
AWS Identity and Access Management

User Guide



AWS Identity and Access Management: User Guide

Copyright © 2018 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What Is IAM?	1
Video Introduction to IAM	1
IAM Features	1
Accessing IAM	2
Understanding How IAM Works	3
Principal	4
Request	4
Authentication	5
Authorization	5
Actions	5
Resources	6
Overview: Users	6
First-Time Access Only: Your Root User Credentials	6
IAM Users	6
Federating Existing Users	7
Overview: Permissions and Policies	8
Policies and Users	8
Policies and Groups	9
Federated Users and Roles	10
User-based and Resource-based Policies	10
Security Features Outside of IAM	11
Quick Links to Common Tasks	12
Getting Set Up	14
Using IAM to Give Users Access to Your AWS Resources	14
Do I Need to Sign Up for IAM?	15
Additional Resources	15
Getting Started	16
Creating an IAM Admin User and Group	17
Creating an Administrator IAM User and Group (Console)	17
Creating an IAM User and Group (AWS CLI)	18
Related Resources	20
How Users Sign In to Your Account	20
Tutorials	22
Tutorial: Delegate Access to the Billing Console	22
Prerequisites	22
Step 1: Enable Access to Billing Data on Your AWS Test Account	23
Step 2: Create IAM Policies That Grant Permissions to Billing Data	23
Step 3: Attach Billing Policies to Your Groups	24
Step 4: Test Access to the Billing Console	24
Related Resources	25
Summary	26
Tutorial: Delegate Access Across AWS Accounts Using Roles	26
Prerequisites	27
Step 1 - Create a Role	27
Step 2 - Grant Access to the Role	29
Step 3 - Test Access by Switching Roles	31
Related Resources	34
Summary	34
Tutorial: Create a Customer Managed Policy	34
Prerequisites	35
Step 1: Create the Policy	35
Step 2: Attach the Policy	36
Step 3: Test User Access	36
Related Resources	36

Summary	37
Tutorial: Enable Users to Configure Their Own Credentials and MFA Settings	37
Prerequisites	37
Step 1: Create a Policy to Enforce MFA Sign-In	38
Step 2: Attach Policies to Your Test Group	41
Step 3: Test Your User's Access	41
Related Resources	42
Best Practices and Use Cases	43
Best Practices	43
Lock Away Your AWS Account Root User Access Keys	43
Create Individual IAM Users	44
Use AWS Defined Policies to Assign Permissions Whenever Possible	44
Use Groups to Assign Permissions to IAM Users	44
Grant Least Privilege	45
Use Access Levels to Review IAM Permissions	45
Configure a Strong Password Policy for Your Users	46
Enable MFA for Privileged Users	46
Use Roles for Applications That Run on Amazon EC2 Instances	46
Delegate by Using Roles Instead of by Sharing Credentials	47
Rotate Credentials Regularly	47
Remove Unnecessary Credentials	47
Use Policy Conditions for Extra Security	47
Monitor Activity in Your AWS Account	48
Video Presentation About IAM Best Practices	48
Business Use Cases	49
Initial Setup of Example Corp	49
Use Case for IAM with Amazon EC2	49
Use Case for IAM with Amazon S3	50
IAM Console and Sign-in Page	52
The IAM User Sign-in Page	52
The AWS Account Root User Sign-in Page	53
Controlling User Access to the AWS Management Console	53
Your AWS Account ID and Its Alias	54
Finding Your AWS Account ID	54
About Account Aliases	55
Creating, Deleting, and Listing an AWS Account Alias	55
Using MFA Devices With Your IAM Sign-in Page	56
IAM Console Search	56
Using IAM Console Search	57
Icons in the IAM Console Search Results	57
Sample Search Phrases	58
Identities	59
The AWS Account Root User	59
IAM Users	59
IAM Groups	59
IAM Roles	60
Temporary Credentials	60
When to Create an IAM User (Instead of a Role)	60
When to Create an IAM Role (Instead of a User)	60
Users	61
How AWS identifies an IAM user	61
Users and credentials	62
Users and permissions	62
Users and accounts	63
Users as service accounts	63
Adding a User	63
How IAM Users Sign In to AWS	67

Managing Users	68
Changing Permissions for a User	71
Passwords	74
Access Keys	85
Retrieving Lost Passwords or Access Keys	90
Multi-Factor Authentication (MFA)	91
Finding Unused Credentials	117
Getting Credential Reports	120
Using IAM with AWS CodeCommit: Git Credentials, SSH Keys, and AWS Access Keys	123
Working with Server Certificates	125
Groups	129
Creating Groups	131
Managing Groups	131
Roles	135
Terms and Concepts	135
Common Scenarios	137
Identity Providers and Federation	143
Service-Linked Roles	176
Creating Roles	184
Using Roles	205
Managing Roles	221
Roles vs. Resource-based Policies	229
Temporary Security Credentials	231
AWS STS and AWS Regions	232
Common Scenarios for Temporary Credentials	232
Requesting Temporary Security Credentials	233
Using Temporary Security Credentials to Request Access to AWS Resources	242
Controlling Permissions for Temporary Security Credentials	246
Activating and Deactivating AWS STS in an AWS Region	257
Sample Applications That Use Temporary Credentials	259
Additional Resources for Temporary Credentials	260
The Root User	260
Enable MFA on the AWS Account Root User	261
Creating Access Keys for the Root User	261
Deleting Access Keys from the Root User	262
Changing the Root User's Password	262
Log Events with CloudTrail	262
Types of IAM Information Logged in CloudTrail	263
Examples of Logged Events in CloudTrail Files	265
Preventing Duplicate Log Entries in CloudTrail	271
Access Management	274
Access Management Resources	275
Policies	275
Identity-Based Policies	275
Resource-Based Policies	276
Overview of JSON Policies	276
Managed Policies and Inline Policies	279
Identity vs Resource	285
Control Access Using Policies	287
Example Policies	295
Managing IAM Policies	316
Creating IAM Policies	317
Validating JSON Policies	321
Testing IAM Policies	322
Attaching and Detaching IAM Policies	331
Versioning IAM Policies	335
Editing IAM Policies	337

Deleting IAM Policies	341
Reduce Policy Scope	342
Understanding Policies	347
Policy Summary (List of Services)	348
Service Summary (List of Actions)	358
Action Summary (List of Resources)	362
Example Policy Summaries	365
Permissions Required	372
Permissions for Administering IAM Identities	373
Permissions for Working in the AWS Management Console	374
Granting Permissions Across AWS Accounts	374
Permissions for One Service to Access Another	375
Required Actions	375
Example Policies for IAM	376
Troubleshooting IAM	383
Troubleshooting General Issues	383
I lost my access keys	383
I need to access an old account	383
I get "access denied" when I make a request to an AWS service	384
I get "access denied" when I make a request with temporary security credentials	384
Policy variables aren't working	385
Changes that I make are not always immediately visible	385
Troubleshoot Policies	386
Troubleshoot Using the Visual Editor	387
Troubleshoot Using Policy Summaries	390
Troubleshoot Policy Management	396
Troubleshoot JSON Policy Documents	396
Troubleshooting IAM Roles	399
I Can't Assume a Role	400
A New Role Appeared in My AWS Account	400
I Can't Edit or Delete a Role in My AWS Account	401
Troubleshooting Amazon EC2 and IAM	401
When attempting to launch an instance, I don't see the role I expected to see in the Amazon EC2 console IAM role list	401
The credentials on my instance are for the wrong role	402
When I attempt to call the <code>AddRoleToInstanceProfile</code> , I get an <code>AccessDenied</code> error	402
Amazon EC2: When I attempt to launch an instance with a role, I get an <code>AccessDenied</code> error ..	402
I can't access the temporary security credentials on my EC2 instance	403
What do the errors from the <code>info</code> document in the IAM subtree mean?	403
Troubleshooting Amazon S3 and IAM	404
How do I grant anonymous access to an Amazon S3 bucket?	404
I'm signed in as an AWS account root user, why can't I access an Amazon S3 bucket under my account?	404
Troubleshooting SAML 2.0 Federation with AWS	405
Invalid SAML response	405
RoleSessionName is required	405
Not authorized for <code>AssumeRoleWithSAML</code>	406
Invalid RoleSessionName characters	406
Invalid response signature	406
Failed to assume role	407
Could not parse metadata	407
How to View a SAML Response in Your Browser for Troubleshooting	407
Reference	410
IAM Identifiers	410
Friendly Names and Paths	410
IAM ARNs	410
Unique IDs	413

Limits	414
IAM Entity Name Limits	414
IAM Entity Object Limits	415
IAM Entity Character Limits	416
Services That Work with IAM	417
Compute	418
Storage & Migration	419
Database	419
Networking & Content Delivery	419
Developer Tools	420
Management Tools	420
Media	421
Machine Learning	421
Analytics	422
Security, Identity, & Compliance	422
Mobile	423
Application Integration	423
Business Productivity	424
Desktop & App Streaming	424
Internet of Things	424
Game Development	425
Software	425
Additional Resources	425
Policy Reference	425
JSON Element Reference	426
IAM JSON Policy Evaluation Logic	458
Policy Grammar	463
AWS Managed Policies for Job Functions	468
Global & IAM Condition Keys	476
Actions and Context Keys for IAM Policies	485
Actions Grouped by Access Level	615
Resources	715
Users and Groups	715
Credentials (Passwords, Access Keys, and MFA devices)	715
Permissions and Policies	715
Federation and Delegation	716
IAM and Other AWS Products	716
Using IAM with Amazon EC2	716
Using IAM with Amazon S3	716
Using IAM with Amazon RDS	717
Using IAM with Amazon DynamoDB	717
General Security Practices	717
General Resources	717
Making Query Requests	719
Endpoints	719
HTTPS Required	720
Signing IAM API Requests	720
AWS Glossary	721

What Is IAM?

 Follow us on Twitter

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the [best practice of using the root user only to create your first IAM user](#). Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

Topics

- [Video Introduction to IAM \(p. 1\)](#)
- [IAM Features \(p. 1\)](#)
- [Accessing IAM \(p. 2\)](#)
- [Understanding How IAM Works \(p. 3\)](#)
- [Overview of Identity Management: Users \(p. 6\)](#)
- [Overview of Access Management: Permissions and Policies \(p. 8\)](#)
- [Security Features Outside of IAM \(p. 11\)](#)
- [Quick Links to Common Tasks \(p. 12\)](#)

Video Introduction to IAM

AWS Training and Certification provides a 10-minute video introduction to IAM:

[Introduction to AWS Identity and Access Management](#)

IAM Features

IAM gives you the following features:

Shared access to your AWS account

You can grant other people permission to administer and use resources in your AWS account without having to share your password or access key.

Granular permissions

You can grant different permissions to different people for different resources. For example, you might allow some users complete access to Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB, Amazon Redshift, and other AWS services.

For other users, you can allow read-only access to just some S3 buckets, or permission to administer just some EC2 instances, or to access your billing information but nothing else.

Secure access to AWS resources for applications that run on Amazon EC2

You can use IAM features to securely give applications that run on EC2 instances the credentials that they need in order to access other AWS resources. Examples include S3 buckets and RDS or DynamoDB databases.

Multi-factor authentication (MFA)

You can add two-factor authentication to your account and to individual users for extra security. With MFA you or your users must provide not only a password or access key to work with your account, but also a code from a specially configured device.

Identity federation

You can allow users who already have passwords elsewhere—for example, in your corporate network or with an internet identity provider—to get temporary access to your AWS account.

Identity information for assurance

If you use [AWS CloudTrail](#), you receive log records that include information about those who made requests for resources in your account. That information is based on IAM identities.

PCI DSS Compliance

IAM supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard (DSS). For more information about PCI DSS, including how to request a copy of the AWS PCI Compliance Package, see [PCI DSS Level 1](#).

Integrated with many AWS services

For a list of AWS services that work with IAM, see [AWS Services That Work with IAM \(p. 417\)](#).

Eventually Consistent

IAM, like many other AWS services, is [eventually consistent](#). IAM achieves high availability by replicating data across multiple servers within Amazon's data centers around the world. If a request to change some data is successful, the change is committed and safely stored. However, the change must be replicated across IAM, which can take some time. Such changes include creating or updating users, groups, roles, or policies. We recommend that you do not include such IAM changes in the critical, high-availability code paths of your application. Instead, make IAM changes in a separate initialization or setup routine that you run less frequently. Also, be sure to verify that the changes have been propagated before production workflows depend on them. For more information, see [Changes that I make are not always immediately visible \(p. 385\)](#).

Free to use

AWS Identity and Access Management is a feature of your AWS account offered at no additional charge. You will be charged only for use of other AWS products by your IAM users. For information about the pricing of other AWS products, see the [Amazon Web Services pricing page](#).

AWS Security Token Service is an included feature of your AWS account offered at no additional charge. You are charged only for the use of other AWS services that are accessed by your AWS STS temporary security credentials. For information about the pricing of other AWS services, see the [Amazon Web Services pricing page](#).

Accessing IAM

You can work with AWS Identity and Access Management in any of the following ways.

AWS Management Console

The console is a browser-based interface to manage IAM and AWS resources. For more information about accessing IAM through the console, see [The IAM Console and Sign-in Page \(p. 52\)](#). For a tutorial that guides you through using the console, see [Creating Your First IAM Admin User and Group \(p. 17\)](#).

AWS Command Line Tools

You can use the AWS command line tools to issue commands at your system's command line to perform IAM and AWS tasks. Using the command line can be faster and more convenient than the console. The command line tools are also useful if you want to build scripts that perform AWS tasks.

AWS provides two sets of command line tools: the [AWS Command Line Interface \(AWS CLI\)](#) and the [AWS Tools for Windows PowerShell](#). For information about installing and using the AWS CLI, see the [AWS Command Line Interface User Guide](#). For information about installing and using the Tools for Windows PowerShell, see the [AWS Tools for Windows PowerShell User Guide](#).

AWS SDKs

AWS provides SDKs (software development kits) that consist of libraries and sample code for various programming languages and platforms (Java, Python, Ruby, .NET, iOS, Android, etc.). The SDKs provide a convenient way to create programmatic access to IAM 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 the [Tools for Amazon Web Services](#) page.

IAM HTTPS API

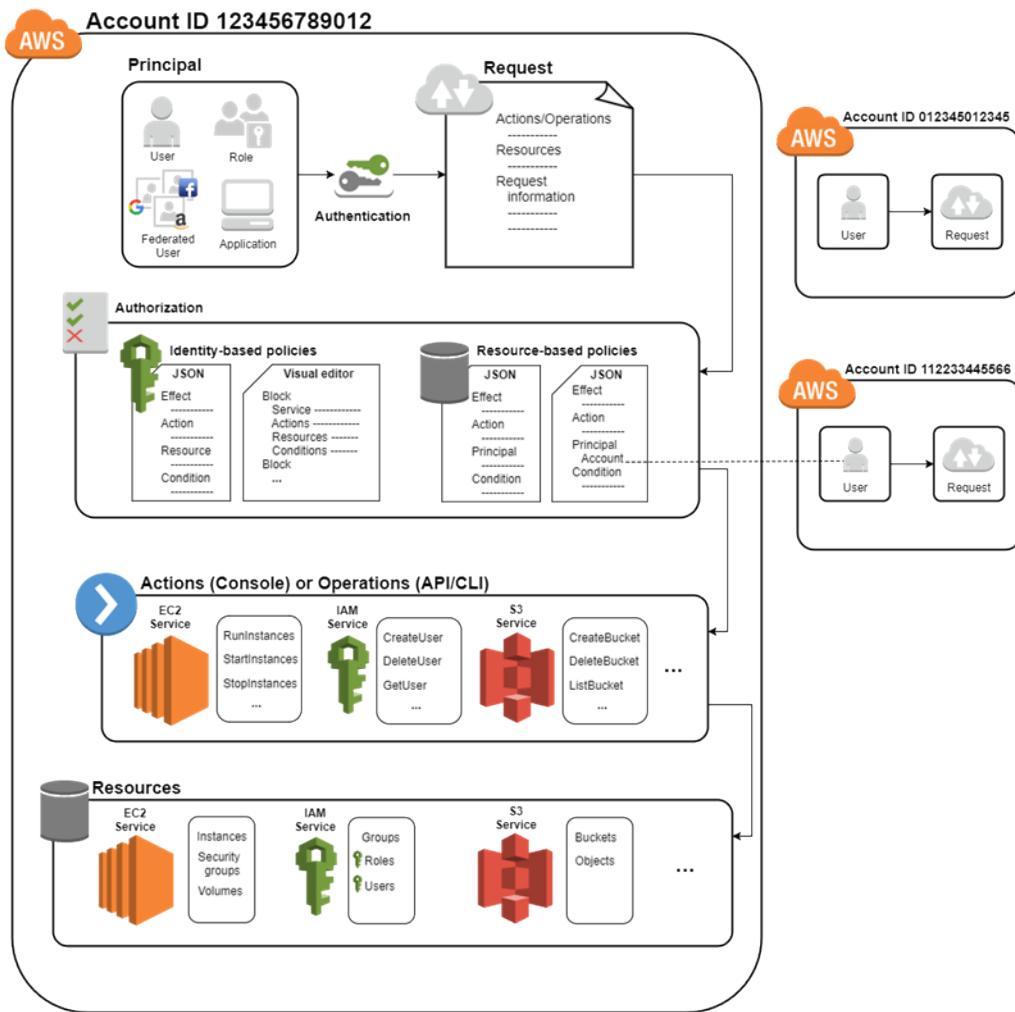
You can access IAM and AWS programmatically by using the IAM HTTPS API, which lets you issue HTTPS requests directly to the service. When you use the HTTPS API, you must include code to digitally sign requests using your credentials. For more information, see [Calling the API by Making HTTP Query Requests \(p. 719\)](#) and the [IAM API Reference](#).

Understanding How IAM Works

Before you create users, you should understand how IAM works. IAM provides the infrastructure necessary to control authentication and authorization for your account. The IAM infrastructure includes the following elements:

Topics

- [Principal \(p. 4\)](#)
- [Request \(p. 4\)](#)
- [Authentication \(p. 5\)](#)
- [Authorization \(p. 5\)](#)
- [Actions \(p. 5\)](#)
- [Resources \(p. 6\)](#)



Principal

A principal is an entity that can take an action on an AWS resource. Your administrative IAM user is your first *principal*. Over time, you can allow users and services to assume a role. You can support federated users or programmatic access to allow an application to access your AWS account. Users, roles, federated users, and applications are all AWS principals.

Request

When a principal tries to use the AWS Management Console, the AWS API, or the AWS CLI, that principal sends a *request* to AWS. A request specifies the following information:

- Actions (or operations) that the principal wants to perform
- Resources upon which the actions are performed
- Principal information, including the environment from which the request was made

Request information is assembled from several sources:

- Principal (the requester), which is determined based on the authorization data. This includes the aggregate permissions that are associated with that principal.
- Environment data, such as the IP address, user agent, SSL enabled status, or the time of day. This information is determined from the request.
- Resource data, or data related to the resource that is being requested. This can include information such as a DynamoDB table name or a tag on an Amazon EC2 instance. This information is determined from the request.

AWS gathers this information into a *request context*, which is used to evaluate and authorize the request.

Authentication

As a principal, you must be authenticated (signed in to AWS) to send a request to AWS. Alternatively, a few services, like Amazon S3, allow requests from anonymous users. To authenticate from the console, you must sign in with your user name and password. To authenticate from the API or CLI, you must provide your access key and secret key. You might also be required to provide additional security information. AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more about the IAM identities that AWS can authenticate, see [Identities \(Users, Groups, and Roles\) \(p. 59\)](#).

Authorization

During authorization, IAM uses values from the request context to check for matching policies and determine whether to allow or deny the request. Policies are stored in IAM as JSON documents and specify the permissions that are allowed or denied for principals (*identity-based policies*) or resources (*resource-based policies*).

IAM checks each policy that matches the context of your request. If a single policy includes a denied action, IAM denies the entire request and stops evaluating. This is called an *explicit deny*. Because requests are *denied by default*, IAM authorizes your request only if every part of your request is allowed by the matching policies. The evaluation logic follows these rules:

- By default, all requests are denied.
- An explicit allow overrides this default.
- An explicit deny overrides any allows.

Note

By default, only the [AWS account root user \(p. 260\)](#) has access to all the resources in that account. So if you are not signed in as the root user, you must have permissions granted by a policy.

Actions

After your request has been authenticated and authorized, AWS approves the actions in your request. Actions are defined by a service, and are the things that you can do to a resource, such as viewing, creating, editing, and deleting that resource. For example, IAM supports around 40 actions for a user resource, including the following actions:

- `CreateUser`
- `DeleteUser`
- `GetUser`
- `UpdateUser`

To allow a principal to perform an action, you must include the necessary actions in a policy that applies to the principal or the affected resource.

Resources

After AWS approves the actions in your request, those actions can be performed on the related resources within your account. A resource is an entity that exists within a service. Examples include an Amazon EC2 instance, an IAM user, and an Amazon S3 bucket. The service defines a set of actions that can be performed on each resource. If you create a request to perform an unrelated action on a resource, that request is denied. For example, if you request to delete an IAM role but provide an IAM group resource, the request fails.

When you provide permissions using an identity-based policy in IAM, then you provide permissions to access resources only within the same account. If you need to make a request in a different account, the resource in that account must have an attached resource-based policy that allows access from your account. Otherwise, you must assume a role within that account with the permissions that you need. To learn more about cross-account permissions, see [Granting Permissions Across AWS Accounts \(p. 374\)](#).

Overview of Identity Management: Users

For greater security and organization, you can give access to your AWS account to specific users—identities that you create with custom permissions. You can further simplify access for those users by federating existing identities into AWS.

Topics

- [First-Time Access Only: Your Root User Credentials \(p. 6\)](#)
- [IAM Users \(p. 6\)](#)
- [Federating Existing Users \(p. 7\)](#)

First-Time Access Only: Your Root User Credentials

When you create an AWS account, you create an AWS account root user identity, which you use to sign in to AWS. You can sign in to the AWS Management Console using this root user identity—that is, the email address and password that you provided when creating the account. This combination of your email address and password is also called your *root user credentials*.

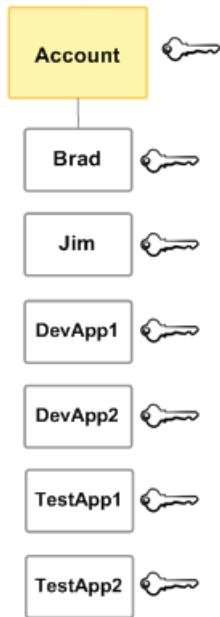
When you use your root user credentials, you have complete, unrestricted access to all resources in your AWS account, including access to your billing information and the ability to change your password. This level of access is necessary when you first set up your account. However, we recommend that you **don't** use root user credentials for everyday access. We especially recommend that you do not share your root user credentials with anyone, because doing so gives them unrestricted access to your account. It is not possible to restrict the permissions that are granted to the root user.

The following sections explain how you can use IAM to create and manage user identity and permissions to provide secure, limited access to your AWS resources, both for yourself and for others who need to work with your AWS resources.

IAM Users

The "identity" aspect of AWS Identity and Access Management (IAM) helps you with the question "Who is that user?", often referred to as *authentication*. Instead of sharing your root user credentials with others, you can create individual IAM users within your account that correspond to users in your organization. IAM users are not separate accounts; they are users within your account. Each user can have its own password for access to the AWS Management Console. You can also create an individual access key for

each user so that the user can make programmatic requests to work with resources in your account. In the following figure, the users Brad, Jim, DevApp1, DevApp2, TestApp1, and TestApp2 have been added to a single AWS account. Each user has its own credentials.



Notice that some of the users are actually applications (for example, DevApp1). An IAM user doesn't have to represent an actual person; you can create an IAM user in order to generate an access key for an application that runs in your corporate network and needs AWS access.

We recommend that you create an IAM user for yourself and then assign yourself administrative permissions for your account. You can then sign in as that user to add more users as needed.

Federating Existing Users

If your users already have a way to be authenticated—for example, by signing in to your corporate network—you can *federate* those user identities into AWS. A user who has already logged in replaces his or her existing identity with a temporary identity in your AWS account. This user can work in the AWS Management Console. Similarly, an application that the user is working with can make programmatic requests using permissions that you define.

Federation is particularly useful in these cases:

- **Your users already have identities in a corporate directory.**

If your corporate directory is compatible with Security Assertion Markup Language 2.0 (SAML 2.0), you can configure your corporate directory to provide single-sign on (SSO) access to the AWS Management Console for your users. For more information, see [Common Scenarios for Temporary Credentials \(p. 232\)](#).

If your corporate directory is not compatible with SAML 2.0, you can create an identity broker application to provide single-sign on (SSO) access to the AWS Management Console for your users. For more information, see [Creating a URL that Enables Federated Users to Access the AWS Management Console \(Custom Federation Broker\) \(p. 169\)](#).

If your corporate directory is Microsoft Active Directory, you can use [AWS Directory Service](#) to establish trust between your corporate directory and your AWS account.

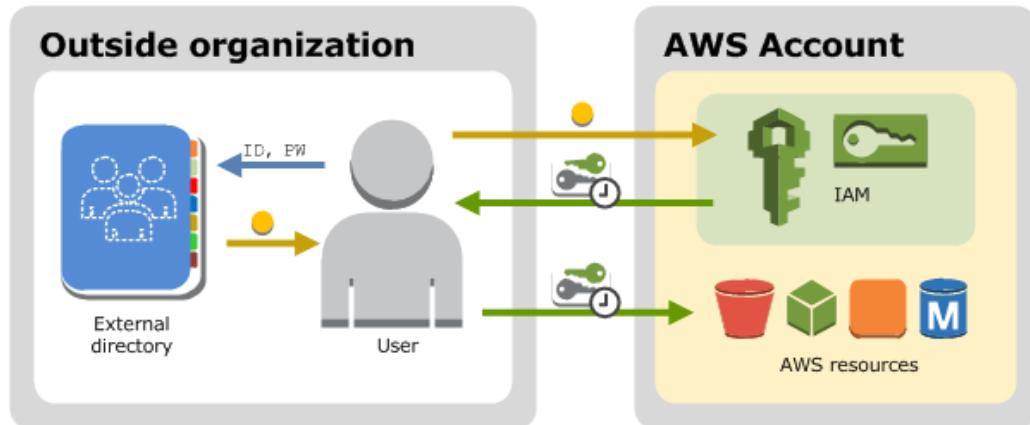
- **Your users already have Internet identities.**

If you are creating a mobile app or web-based app that can let users identify themselves through an Internet identity provider like Login with Amazon, Facebook, Google, or any OpenID Connect (OIDC) compatible identity provider, the app can use federation to access AWS. For more information, see [About Web Identity Federation \(p. 143\)](#).

Tip

To use identity federation with Internet identity providers, we recommend you use [Amazon Cognito](#).

The following diagram shows how a user can use IAM to get temporary AWS security credentials to access resources in your AWS account.



Overview of Access Management: Permissions and Policies

The access management portion of AWS Identity and Access Management (IAM) helps you to define what a user or other entity is allowed to do in an account, often referred to as *authorization*. Permissions are granted through policies that are created and then attached to users, groups, or roles.

Policies and Users

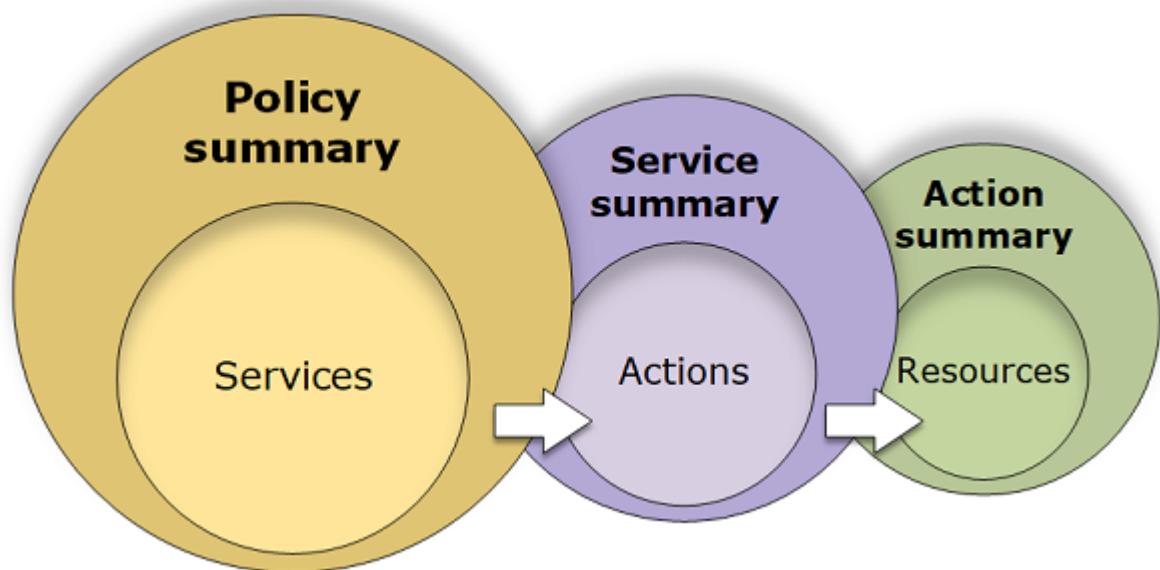
By default, IAM users can't access anything in your account. You grant permissions to a user by creating a *policy*, which is a document that defines the effect, actions, resources, and optional conditions. The following example shows a policy that grants permission to perform Amazon DynamoDB actions (`dynamodb : *`) on the Books table in the account 123456789012 within the us-east-2 region.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "dynamodb:*",  
            "Resource": "arn:aws:dynamodb:us-east-2:123456789012:table/Books"  
        }  
    ]  
}
```

When you attach the policy to a user, group, or role, then the IAM entity has those DynamoDB permissions. Typically, users in your account have multiple policies that together represent the permissions for that user.

Any actions or resources that are not explicitly allowed are denied by default. For example, if the above policy is the only policy attached to a user, then that user is allowed to perform DynamoDB actions on the Books table only. Actions on all other tables are prohibited. Similarly, the user is not allowed to perform any actions in Amazon EC2, Amazon S3, or in any other AWS service, because permissions to work with those services are not included in the policy.

The IAM console includes *policy summary* tables that describe the access level, resources, and conditions that are allowed or denied for each service in a policy. Policies are summarized in three tables: the [policy summary \(p. 348\)](#), the [service summary \(p. 358\)](#), and the [action summary \(p. 362\)](#). The *policy summary* table includes a list of services. Choose a service there to see the *service summary*. This summary table includes a list of the actions and associated permissions for the chosen service. You can choose an action from that table to view the *action summary*. This table includes a list of resources and conditions for the chosen action.



You can view policy summaries on the **Users** page for all policies (managed and inline) that are attached to that user. View summaries on the **Policies** page for all managed policies.

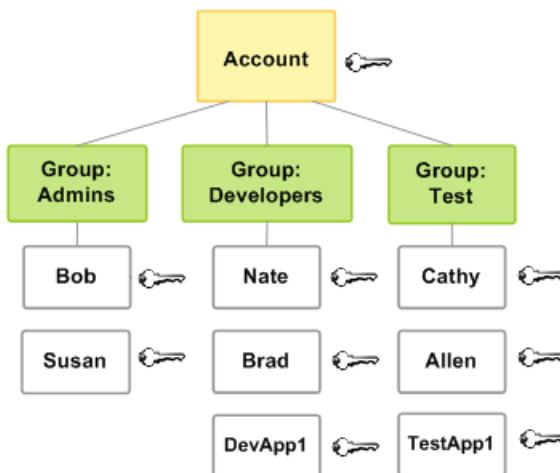
For example, the previous policy is summarized in the AWS Management Console as follows:

Service	Access level	Resource	Request condition
Allow (1 of 102 services) Show remaining 101			
DynamoDB	Full access	TableName = Books	None

You can also view the JSON document for the policy. For information about viewing the summary or JSON document, see [Understanding Permissions Granted by a Policy \(p. 347\)](#).

Policies and Groups

You can organize IAM users into *IAM groups* and attach a policy to a group. In that case, individual users still have their own credentials, but all the users in a group have the permissions that are attached to the group. Use groups for easier permissions management, and to follow our [IAM Best Practices \(p. 43\)](#).



Users or groups can have multiple policies attached to them that grant different permissions. In that case, the users' permissions are calculated based on the combination of policies. But the basic principle still applies: If the user has not been granted an explicit permission for an action and a resource, the user does not have those permissions.

Federated Users and Roles

Federated users don't have permanent identities in your AWS account the way that IAM users do. To assign permissions to federated users, you can create an entity referred to as a *role* and define permissions for the role. When a federated user signs in to AWS, the user is associated with the role and is granted the permissions that are defined in the role. For more information, see [Creating a Role for a Third-Party Identity Provider \(Federation\) \(p. 194\)](#).

User-based and Resource-based Policies

In the previous example, you saw a policy that you can attach to a user or to a group. When you create a policy for a user like that, you specify the actions that are permitted and the resource (EC2 instance, RDS database, etc.) that the user is allowed to access.

In some cases you can attach a policy to a resource in addition to attaching it to a user or group. For example, in Amazon S3, you can attach a policy to a bucket. A *resource-based policy* contains slightly different information than a user-based policy. In a resource-based policy you specify what actions are permitted and what resource is affected (just like a user-based policy). However, you also explicitly list who is allowed access to the resource. (In a user-based policy, the "who" is established by whomever the policy is attached to.)

The following example shows an S3 bucket policy that allows an IAM user named bob in AWS account 777788889999 to put objects into the bucket called example-bucket.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "arn:aws:iam::777788889999:user/bob"},
    "Action": [
      "s3:PutObject",
      "s3:PutObjectAcl"
    ],
    "Resource": "arn:aws:s3:::example-bucket/*"
  }
}
```

```
}
```

Resource-based policies include a `Principal` element that specifies who is granted the permissions. In the preceding example, the `Principal` element is set to the [Amazon Resource Name \(ARN\)](#) of an IAM user named bob in AWS account 777788889999. This indicates that the resource (in this case, the S3 bucket) is accessible to that IAM user but no one else.

Security Features Outside of IAM

You use IAM to control access to tasks that are performed using the AWS Management Console, the [AWS Command Line Tools](#), or service APIs using the [AWS SDKs](#). Some AWS products have other ways to secure their resources as well. The following list provides some examples, though it is not exhaustive.

Amazon EC2

In Amazon Elastic Compute Cloud you log into an instance with a key pair (for Linux instances) or using a user name and password (for Microsoft Windows instances).

For more information, see the following documentation:

- [Getting Started with Amazon EC2 Linux Instances](#) in the *Amazon EC2 User Guide for Linux Instances*
- [Getting Started with Amazon EC2 Windows Instances](#) in the *Amazon EC2 User Guide for Windows Instances*

Amazon RDS

In Amazon Relational Database Service you log into the database engine with a user name and password that are tied to that database.

For more information, see [Getting Started with Amazon RDS](#) in the *Amazon Relational Database Service User Guide*.

Amazon EC2 and Amazon RDS

In Amazon EC2 and Amazon RDS you use security groups to control traffic to an instance or database.

For more information, see the following documentation:

- [Amazon EC2 Security Groups for Linux Instances](#) in the *Amazon EC2 User Guide for Linux Instances*
- [Amazon EC2 Security Groups for Windows Instances](#) in the *Amazon EC2 User Guide for Windows Instances*
- [Amazon RDS Security Groups](#) in the *Amazon Relational Database Service User Guide*

Amazon WorkSpaces

In Amazon WorkSpaces, users sign in to a desktop with a user name and password.

For more information, see [Getting Started with Amazon WorkSpaces](#) in the *Amazon WorkSpaces Administration Guide*.

Amazon WorkDocs

In Amazon WorkDocs, users get access to shared documents by signing in with a user name and password.

For more information, see [Getting Started with Amazon WorkDocs](#) in the *Amazon WorkDocs Administration Guide*.

These access control methods are not part of IAM. IAM lets you control how these AWS products are administered—creating or terminating an Amazon EC2 instance, setting up new Amazon WorkSpaces desktops, and so on. That is, IAM helps you control the tasks that are performed by making requests to Amazon Web Services, and it helps you control access to the AWS Management Console. However, IAM does not help you manage security for tasks like signing in to an operating system (Amazon EC2), database (Amazon RDS), desktop (Amazon WorkSpaces), or collaboration site (Amazon WorkDocs).

When you work with a specific AWS product, be sure to read the documentation to learn the security options for all the resources that belong to that product.

Quick Links to Common Tasks

Use the following links to get help with common tasks associated with IAM.

Sign in as an IAM user

See [How IAM Users Sign In to AWS \(p. 67\)](#).

Manage passwords for IAM users

You need a password in order to access the AWS Management Console, including access to billing information.

For your AWS account root user, see [Changing the AWS Account Root User Password \(p. 75\)](#).

For an IAM user, see [Managing Passwords for IAM Users \(p. 79\)](#).

Manage permissions for IAM users

You use policies to grant permissions to the IAM users in your AWS account. IAM users have no permissions when they are created, so you must add permissions to allow them to use AWS resources.

For more information, see [Managing IAM Policies \(p. 316\)](#).

List the users in your AWS account and get information about their credentials

See [Getting Credential Reports for Your AWS Account \(p. 120\)](#).

Add multi-factor authentication (MFA)

To add a virtual MFA device for your AWS account root user, see [Enable a Virtual MFA Device for Your AWS Account Root User \(Console\) \(p. 94\)](#).

To add a hardware MFA device for your root user, see [Enable a Hardware MFA Device for the AWS Account Root User \(Console\) \(p. 97\)](#).

To add a virtual MFA device for an IAM user, see [Enable a Virtual MFA Device for an IAM User \(AWS Management Console\) \(p. 93\)](#).

To add a hardware MFA device for an IAM user, see [Enable a Hardware MFA Device for an IAM User \(Console\) \(p. 96\)](#).

To add a hardware MFA device for your AWS account or an IAM user, see [Enabling a Hardware MFA Device \(Console\) \(p. 96\)](#).

Get an access key

You need an access key if you want to make AWS requests using the AWS SDKs, the [AWS Command Line Tools](#), [Tools for Windows PowerShell](#) or the APIs.

Important

You can view and download your secret access key *only* when you create the access key. You cannot view or recover a secret access key later. However, if you lose your secret access key, you can create a new access key.

For your AWS account, see [Managing Access Keys for your AWS Account](#).

For an IAM user, see [Managing Access Keys for IAM Users \(p. 85\)](#).

Get started with all of AWS

This set of documentation deals primarily with the IAM service. To learn about getting started with AWS and using multiple services to solve a problem such as building and launching your first project, see the [Getting Started Resource Center](#).

Getting Set Up

AWS Identity and Access Management (IAM) helps you securely control access to Amazon Web Services (AWS) and your account resources. IAM can also keep your account credentials private. With IAM, you can create multiple IAM users under the umbrella of your AWS account or enable temporary access through identity federation with your corporate directory. In some cases, you can also enable access to resources across AWS accounts.

Without IAM, however, you must either create multiple AWS accounts—each with its own billing and subscriptions to AWS products—or your employees must share the security credentials of a single AWS account. In addition, without IAM, you cannot control the tasks a particular user or system can do and what AWS resources they might use.

This guide provides a conceptual overview of IAM, describes business use cases, and explains AWS permissions and policies.

Topics

- [Using IAM to Give Users Access to Your AWS Resources \(p. 14\)](#)
- [Do I Need to Sign Up for IAM? \(p. 15\)](#)
- [Additional Resources \(p. 15\)](#)

Using IAM to Give Users Access to Your AWS Resources

Here are the ways you can use IAM to control access to your AWS resources.

Type of access	Why would I use it?	Where can I get more information?
Access for users under your AWS account	You want to add users under the umbrella of your AWS account, and you want to use IAM to create users and manage their permissions.	To learn how to use the AWS Management Console to create users and to manage their permissions under your AWS account, see Getting Started (p. 16) . To learn about using the IAM API or AWS Command Line Interface to create users under your AWS account, see Creating Your First IAM Admin User and Group (p. 17) . For more information about working with IAM users, see Identities (Users, Groups, and Roles) (p. 59) .
Non-AWS user access via identity federation between your authorization system and AWS	You have non-AWS users in your identity and authorization system, and they need access to your AWS resources.	To learn how to use security tokens to give your users access to your AWS account resources through federation with your corporate directory, go to Temporary Security Credentials (p. 231) . For information about the AWS Security Token Service API, go to the AWS Security Token Service API Reference .

Type of access	Why would I use it?	Where can I get more information?
Cross-account access between AWS accounts	You want to share access to certain AWS resources with users under other AWS accounts.	To learn how to use IAM to grant permissions to other AWS accounts, see Roles Terms and Concepts (p. 135) .

Do I Need to Sign Up for IAM?

If you don't already have an AWS account, you need to create one to use IAM. You don't need to specifically sign up to use IAM. There is no charge to use IAM.

Note

IAM works only with AWS products that are integrated with IAM. For a list of services that support IAM, see [AWS Services That Work with IAM \(p. 417\)](#).

To sign up for AWS

1. Open <https://aws.amazon.com/>, and then choose **Create an AWS Account**.

Note

This might be unavailable in your browser if you previously signed into the AWS Management Console. In that case, choose **Sign in to a different account**, and then choose **Create a new AWS account**.

2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

Additional Resources

Here are some resources to help you get things done with IAM.

- Manage your AWS account credentials: [AWS Security Credentials](#) in the [AWS General Reference](#)
- Get started with and learn more about [What Is IAM? \(p. 1\)](#)
- Set up a command line interface (CLI) to use with IAM. For the cross-platform AWS CLI, see the [AWS Command Line Interface Documentation](#) and [IAM CLI reference](#). You can also manage IAM with Windows PowerShell; see the [AWS Tools for Windows PowerShell Documentation](#) and [IAM Windows PowerShell reference](#).
- Download an AWS SDK for convenient programmatic access to IAM: [Tools for Amazon Web Services](#)
- Get the FAQ: [AWS Identity and Access Management FAQ](#)
- Get technical support: [AWS Support Center](#)
- Get premium technical support: [AWS Premium Support Center](#)
- Find definitions of AWS terms: [Amazon Web Services Glossary](#)
- Get community support: [IAM Discussion Forums](#)
- Contact AWS: [Contact Us](#)

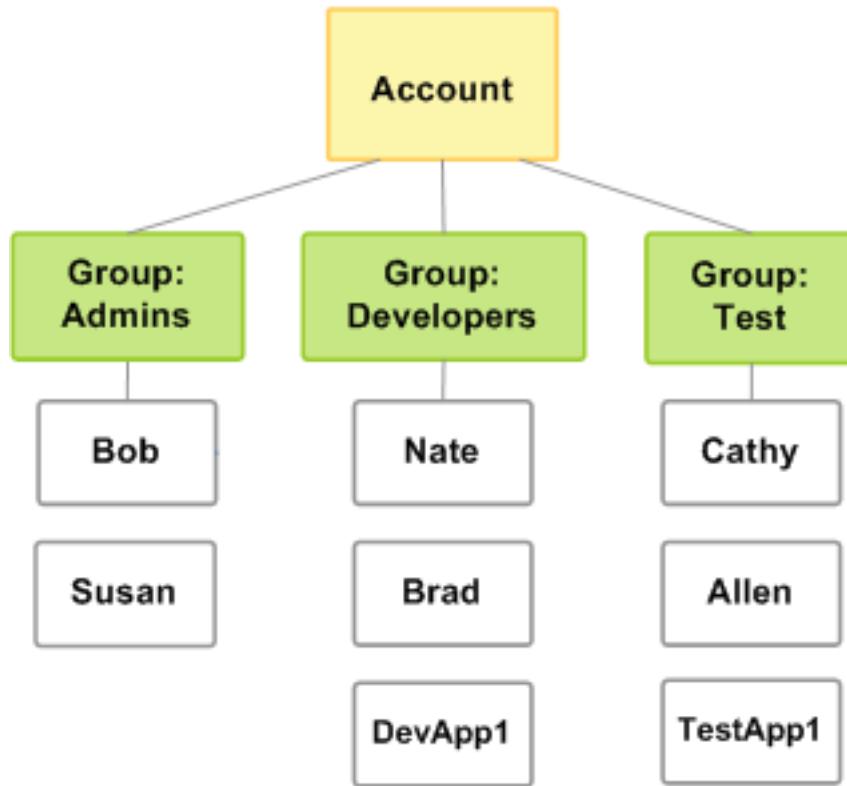
Getting Started

This topic shows you how to give access to your AWS resources by creating AWS Identity and Access Management (IAM) users under your AWS account. First, you'll learn about IAM concepts you should understand before you create groups and users, and then you'll walk through how to perform the necessary tasks using the AWS Management Console. The first task is to set up an administrators group for your AWS account. Having an administrators group for your AWS account isn't required, but we strongly recommend it.

Note

This set of documentation deals primarily with the IAM service. To learn about getting started with AWS and using multiple services to solve a problem such as building and launching your first project, see the [Getting Started Resource Center](#).

The following figure shows a simple example of an AWS account with three groups. A group is a collection of users who have similar responsibilities. In this example, one group is for administrators (it's called *Admins*). There's also a *Developers* group and a *Test* group. Each group has multiple users. Each user can be in more than one group, although the figure doesn't illustrate that. You can't put groups inside other groups. You use policies to grant permissions to groups.



In the procedure that follows, you will perform the following tasks:

- Create an Administrators group and give the group permission to access all of your AWS account's resources.
- Create a user for yourself and add that user to the Administrators group.
- Create a password for your user so you can sign in to the AWS Management Console.

You will grant the Administrators group permission to access all your available AWS account resources. Available resources are any AWS products you use, or that you are signed up for. Users in the Administrators group can also access your AWS account information, *except* for your AWS account's security credentials.

Topics

- [Creating Your First IAM Admin User and Group \(p. 17\)](#)
- [How Users Sign In to Your Account \(p. 20\)](#)

Creating Your First IAM Admin User and Group

Important

If you arrived at this page trying to enable Amazon Advertising for your application or web site, see [Becoming a Product Advertising API Developer](#).

As a [best practice \(p. 43\)](#), do not use the AWS account root user for any task where it's not required. Instead, create a new IAM user for each person that requires administrator access. Then make those users administrators by placing the users into an "Administrators" group to which you attach the AdministratorAccess managed policy.

Thereafter, the users in the administrators group should set up the groups, users, and so on, for the AWS account. All future interaction should be through the AWS account's users and their own keys instead of the root user. However, to perform some account and service management tasks, you must log in using the root user credentials. To view the tasks that require you to sign in as the root user, see [AWS Tasks that Require Account Root User](#).

Creating an Administrator IAM User and Group (Console)

This procedure describes how to use the AWS Management Console to create an IAM user for yourself and add that user to a group that has administrative permissions from an attached managed policy.

To create an administrator user for yourself and add the user to an administrators group (console)

1. Use your AWS account email address and password to sign in as the [AWS account root user](#) to the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Users** and then choose **Add user**.
3. For **User name**, type a user name, such as **Administrator**. The name can consist of letters, digits, and the following characters: plus (+), equal (=), comma (,), period (.), at (@), underscore (_), and hyphen (-). The name is not case sensitive and can be a maximum of 64 characters in length.
4. Select the check box next to **AWS Management Console access**, select **Custom password**, and then type your new password in the text box. If you're creating the user for someone other than yourself, you can optionally select **Require password reset** to force the user to create a new password when first signing in.
5. Choose **Next: Permissions**.
6. On the **Set permissions for user** page, choose **Add user to group**.
7. Choose **Create group**.
8. In the **Create group** dialog box, type the name for the new group. The name can consist of letters, digits, and the following characters: plus (+), equal (=), comma (,), period (.), at (@), underscore (_), and hyphen (-). The name is not case sensitive and can be a maximum of 128 characters in length.
9. In the policy list, select the check box next to **AdministratorAccess**. Then choose **Create group**.

10. Back in the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
11. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose **Create user**.

You can use this same process to create more groups and users and to give your users access to your AWS account resources. To learn about using policies that restrict user permissions to specific AWS resources, see [Access Management \(p. 274\)](#) and [Example Policies \(p. 295\)](#). To add additional users to the group after it's created, see [Adding and Removing Users in an IAM Group \(p. 132\)](#).

Creating an IAM User and Group (AWS CLI)

If you followed the steps in the previous section, you used the AWS Management Console to set up an administrators group while creating the IAM user in your AWS account. This procedure shows an alternative way to create a group.

Overview: Setting Up an Administrators Group

1. Create a group and give it a name (for example, Admins). For more information, see [Creating a Group \(AWS CLI\) \(p. 18\)](#).
2. Attach a policy that gives the group administrative permissions—access to all AWS actions and resources. For more information, see [Attaching a Policy to the Group \(AWS CLI\) \(p. 19\)](#).
3. Add at least one user to the group. For more information, see [Creating an IAM User in Your AWS Account \(p. 63\)](#).

Creating a Group (AWS CLI)

This section shows how to create a group in the IAM system.

To create an administrators group (AWS CLI)

1. Type the `aws iam create-group` command with the name you've chosen for the group. Optionally, you can include a path as part of the group name. For more information about paths, see [Friendly Names and Paths \(p. 410\)](#). The name can consist of letters, digits, and the following characters: plus (+), equal (=), comma (,), period (.), at (@), underscore (_), and hyphen (-). The name is not case sensitive and can be a maximum of 128 characters in length.

In this example, you create a group named Admins.

```
aws iam create-group --group-name Admins
{
    "Group": {
        "Path": "/",
        "CreateDate": "2014-06-05T20:29:53.622Z",
        "GroupId": "ABCDEFGHIJKLMNOPTED",
        "Arn": "arn:aws:iam::123456789012:group/Admins",
        "GroupName": "Admins"
    }
}
```

2. Type the `aws iam list-groups` command to list the groups in your AWS account and confirm the group was created.

```
aws iam list-groups
{
    "Groups": [

```

```
{  
    "Path": "/",
    "CreateDate": "2014-06-05T20:29:53.622Z",
    "GroupId": "ABCDEFGHIJKLMNO",
    "Arn": "arn:aws:iam::123456789012:group/Admins",
    "GroupName": "Admins"
}  
}]  
}
```

The response includes the Amazon Resource Name (ARN) for your new group. The ARN is a standard format that AWS uses to identify resources. The 12-digit number in the ARN is your AWS account ID. The friendly name you assigned to the group (Admins) appears at the end of the group's ARN.

Attaching a Policy to the Group (AWS CLI)

This section shows how to attach a policy that lets any user in the group perform any action on any resource in the AWS account. You do this by attaching the [AWS managed policy \(p. 279\)](#) called AdministratorAccess to the Admins group. For more information about policies, see [Access Management \(p. 274\)](#).

To add a policy giving full administrator permissions (AWS CLI)

1. Type the `aws iam attach-group-policy` command to attach the policy called AdministratorAccess to your Admins group. The command uses the ARN of the AWS managed policy called AdministratorAccess.

```
aws iam attach-group-policy --group-name Admins --policy-arn arn:aws:iam::aws:policy/
AdministratorAccess
```

If the command is successful, there is no response.

2. Type the `aws iam list-attached-group-policies` command to confirm the policy is attached to the Admins group.

```
aws iam list-attached-group-policies --group-name Admins
```

The response lists the names of the policies attached to the Admins group. A response like the following tells you that the policy named AdministratorAccess has been attached to the Admins group:

```
{  
    "AttachedPolicies": [  
        {  
            "PolicyName": "AdministratorAccess",  
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"  
        }  
    ],  
    "IsTruncated": false  
}
```

You can confirm the contents of a particular policy with the `aws iam get-policy` command.

Important

After you have the administrators group set up, you must add at least one user to it. For more information about adding users to a group, see [Creating an IAM User in Your AWS Account \(p. 63\)](#).

Related Resources

For related information found in the *Amazon Web Services General Reference*, see the following resources:

- [AWS Tasks that Require Account Root User](#)

For related information in the *IAM User Guide*, see the following resources:

- [Reducing Policy Scope by Viewing User Activity \(p. 342\)](#)
- [Tutorial: Delegate Access to the Billing Console \(p. 22\)](#)

How Users Sign In to Your Account

After you create IAM users and passwords for each, users can sign in to the AWS Management Console for your AWS account using your account ID or alias, or from a special URL that includes your account ID.

By default, the sign-in URL for your account includes your account ID. You can create a unique sign-in URL for your account so that the URL includes a name instead of an account ID. For more information, see [Your AWS Account ID and Its Alias \(p. 54\)](#).

The sign-in endpoint follows this pattern:

```
https://AWS-account-ID-or-alias.signin.aws.amazon.com/console
```

You can find the sign-in URL for an account on the IAM console dashboard.

IAM users sign-in link:

<https://my-account.signin.aws.amazon.com/console>

[Customize](#) | [Copy Link](#)

You can also sign in at the following endpoint and enter the account ID or alias manually, instead of it being embedded in the URL:

```
https://signin.aws.amazon.com/console
```

Tip

To create a bookmark for your account's unique sign-in page in your web browser, you should manually enter your account's sign-in URL in the bookmark entry. Don't use your web browser's "bookmark this page" feature.

IAM users in your account have access only to the AWS resources that you specify in the policy that is attached to the user or to an IAM group that the user belongs to. To work in the console, users must have permissions to perform the actions that the console performs, such as listing and creating AWS resources. For more information, see [Access Management \(p. 274\)](#) and [Example Policies \(p. 295\)](#).

Note

If your organization has an existing identity system, you might want to create a single sign-on (SSO) option. SSO gives users access to the AWS Management Console for your account without requiring them to have an IAM user identity. SSO also eliminates the need for users to sign in to your organization's site and to AWS separately. For more information, see [Creating a URL that Enables Federated Users to Access the AWS Management Console \(Custom Federation Broker\) \(p. 169\)](#).

Logging sign-in details in CloudTrail

If you enable CloudTrail to log sign-in events to your logs, you need to be aware of how CloudTrail chooses where to log the events.

- If your users sign-in directly to a console, they are redirected to either a global or a regional sign-in endpoint, based on whether the selected service console supports regions. For example, the main console home page supports regions, so if you sign in to the following URL:

```
https://alias.signin.aws.amazon.com/console
```

you are redirected to a "default" regional sign-in endpoint `https://us-east-1.signin.aws.amazon.com`, resulting in a regional CloudTrail log entry in that region's log:

On the other hand, the Amazon S3 console does not support regions, so if you sign in to the following URL

```
https://alias.signin.aws.amazon.com/console/s3
```

AWS redirects you to the global sign-in endpoint at `https://signin.aws.amazon.com`, resulting in a global CloudTrail log entry.

- You can manually request a certain regional sign-in endpoint by signing in to the region-enabled main console home page using a URL syntax like the following:

```
https://alias.signin.aws.amazon.com/console?region=ap-southeast-1
```

AWS redirects you to the `ap-southeast-1` regional sign-in endpoint and results in a CloudTrail log event in that region.

For more information about CloudTrail and IAM, see [Logging IAM Events with AWS CloudTrail](#).

If users need programmatic access to work with your account, you can create an access key pair (an access key ID and a secret access key) for each user, as described in [Creating, Modifying, and Viewing Access Keys \(Console\) \(p. 85\)](#).

IAM Tutorials

This section contains walkthroughs that present complete end-to-end procedures for common tasks that you can perform in IAM. They are intended for a lab-type environment, with sample company names, user names, and so on. Their purpose is to provide general guidance. They are not intended for direct use in your production environment without careful review and adaptation to the unique aspects of your organization's environment.

Topics

- [Tutorial: Delegate Access to the Billing Console \(p. 22\)](#)
- [Tutorial: Delegate Access Across AWS Accounts Using IAM Roles \(p. 26\)](#)
- [Tutorial: Create and Attach Your First Customer Managed Policy \(p. 34\)](#)
- [Tutorial: Enable Your Users to Configure Their Own Credentials and MFA Settings \(p. 37\)](#)

Tutorial: Delegate Access to the Billing Console

AWS account owners can delegate access to specific IAM users who need to view or manage the AWS Billing and Cost Management data for an AWS account. The instructions that follow will help you set up a pretested scenario so you can gain hands-on experience configuring billing permissions, without concern for affecting your main AWS production account.

This workflow has four basic steps.

Step 1: Enable Access to Billing Data on Your AWS Test Account (p. 23)

By default, only the AWS account owner ([AWS account root user \(p. 260\)](#)) has access to view and manage billing information. IAM users cannot access billing data until the account owner provides the user with permission. To view additional tasks that require you to sign in as the root user, see [AWS Tasks that Require Account Root User](#).

Step 2: Create IAM Policies That Grant Permissions to Billing Data (p. 23)

After enabling billing access on your account, you must still explicitly grant access to billing data to specific IAM users or groups. You grant this access with a customer managed policy.

Step 3: Attach Billing Policies to Your Groups (p. 24)

When you attach a policy to a group, all members of that group receive the complete set of access permissions that are associated with that policy. In this scenario, you attach the new billing policies to groups containing only those users who require the billing access.

Step 4: Test Access to the Billing Console (p. 24)

Once you've completed the core tasks, you're ready to test the policy. Testing ensures that the policy works the way you want it to.

Prerequisites

Create a test AWS account to use with this tutorial. In this account create two test users and two test groups as summarized in the following table. Be sure to assign a password to each user so that you can sign in later in Step 4.

Create user accounts	Create and configure group accounts	
FinanceManager	FullAccess	FinanceManager
FinanceUser	ViewAccess	FinanceUser

Step 1: Enable Access to Billing Data on Your AWS Test Account

Sign into your test account and turn on billing access. For information about how to follow this process in a production environment, see [Activate Access to the AWS Website](#) in the *AWS Billing and Cost Management User Guide*.

To enable access to billing data on your AWS test account

1. Use your AWS account email address and password to sign in to the [AWS Management Console](#) as the [AWS account root user](#) (p. 260).
2. On the navigation bar, choose your account name, and then choose **My Account**.
3. Next to **IAM User and Role Access to Billing Information**, choose **Edit**, and then select the check box to activate IAM user and federated user access to the Billing and Cost Management pages.
4. Sign out of the console, and then proceed to [Step 2: Create IAM Policies That Grant Permissions to Billing Data](#) (p. 23).

Step 2: Create IAM Policies That Grant Permissions to Billing Data

Next, create custom policies that grant both view and full access permissions to the pages within the Billing and Cost Management console. For general information about IAM permission policies, see [Managed Policies and Inline Policies](#) (p. 279).

To create IAM policies that grant permissions to billing data

1. Sign in to the AWS Management Console as a user with administrator credentials. To adhere to IAM best practices, don't sign in with your root user credentials. For more information, see [Create individual IAM users](#) (p. 44).
2. Open the IAM console at <https://console.aws.amazon.com/iam/>.
3. In the navigation pane, choose **Policies**, and then choose **Create policy**.
4. On the **Visual editor** tab, choose **Choose a service** to get started. Then choose **Billing**.
5. Follow these steps to create two policies:

Full access

- a. Choose **Select actions** and then select the check box next to **All Actions (*)**. You do not need to select a resource or condition for this policy.
- b. Choose **Review policy**.
- c. On the **Review** page, next to **Name**, type **BillingFullAccess**, and then choose **Create policy** to save it.

Read-only access

- a. Repeat steps [3 and 4 \(p. 23\)](#).
- b. Choose **Select actions** and then select the check box next to **Read**. You do not need to select a resource or condition for this policy.
- c. Choose **Review policy**.
- d. On the **Review** page, for **Name**, type **BillingViewAccess**. Then choose **Create policy** to save it.

To review descriptions for each of the permissions available in IAM policies that grant users access to the Billing and Cost Management console, see [Billing Permissions Descriptions](#).

Step 3: Attach Billing Policies to Your Groups

Now that you have custom billing policies available, you can attach them to their corresponding groups that you created earlier. Although you can attach a policy directly to a user or role, we recommend (in accordance with IAM best practices) that you use groups instead. For more information, see [Use groups to assign permissions to IAM users \(p. 44\)](#).

To attach billing policies to your groups

1. In the navigation pane, choose **Policies** to display the full list of policies available to your AWS account. To attach each policy to its appropriate group, follow these steps:

Full access

- a. In the search box, type **BillingFullAccess**, and then select the check box next to the policy name.
- b. Choose **Policy actions**, and then choose **Attach**.
- c. In the search box, type **FinanceManager**, select the check box next to the name of the group, and then choose **Attach policy**.

Read-only access

- a. In the search box, type **BillingViewAccess**, and then select the check box next to the policy name.
 - b. Choose **Policy actions**, and then choose **Attach**.
 - c. For **Filter**, choose **Groups**. In the search box, type **FinanceUser**, select the check box next to the name of the group, and then choose **Attach policy**.
2. Sign out of the console, and then proceed to [Step 4: Test Access to the Billing Console \(p. 24\)](#).

Step 4: Test Access to the Billing Console

You can test user access in a couple of ways. For this tutorial, we recommend that you test access by signing in as each of the test users so you can see what your users might experience. Another (optional) way to test user access permissions is to use the [IAM policy simulator](#). Use the following steps if you want to see another way to view the effective result of these actions.

Select either of the following procedures based on your preferred testing method. In the first one, you sign in using both test accounts to see the difference between access rights.

To test billing access by signing in with both test user accounts

1. Use your AWS account ID or account alias, your IAM user name, and your password to sign in to the [IAM console](#).

Note

For your convenience, the AWS sign-in page uses a browser cookie to remember your IAM user name and account information. If you previously signed in as a different user, choose **Sign in to a different account** near the bottom of the page to return to the main sign-in page. From there, you can type your AWS account ID or account alias to be redirected to the IAM user sign-in page for your account.

2. Sign-in with each account using the steps provided below so you can compare the different user experiences.

Full access

- a. Sign in to your AWS account as the user FinanceManager.
- b. On the navigation bar, choose **FinanceManager@<account alias or ID number>**, and then choose **Billing & Cost Management**.
- c. Browse through the pages and choose the various buttons to ensure that you have full modify permissions.

Read-only access

- a. Sign in to your AWS account as the user FinanceUser.
- b. On the navigation bar, choose **FinanceUser@<account alias or ID number>**, and then choose **Billing & Cost Management**.
- c. Browse through the pages. Notice that you can display costs, reports, and billing data with no problems. However, if you choose an option to modify a value, you receive an **Access Denied** message. For example, on the **Preferences** page, choose any of the check boxes on the page, and then choose **Save preferences**. The console message informs you that you need **ModifyBilling** permissions to make changes to that page.

The following optional procedure demonstrates how you could alternatively use the IAM policy simulator to test your delegated user's effective permissions to billing pages.

To test billing access by viewing effective permissions in the IAM policy simulator

1. Open the IAM policy simulator at <https://policysim.aws.amazon.com/>. (If you are not already signed in to AWS, you are prompted to sign in).
2. Under **Users, Groups, and Roles**, select one of the users that is a member of the group you recently attached the policy to.
3. Under **Policy Simulator**, choose **Select service**, and then choose **Billing**.
4. Next to **Select actions**, choose **Select All**.
5. Choose **Run Simulation** and compare the user's listed permissions with all possible billing-related permission options to make sure that the correct rights have been applied.

Related Resources

For related information found in the *AWS Billing and Cost Management User Guide*, see the following resources:

- [Activate Access to the AWS Website](#)
- [Example 4: Allow full access to AWS services but deny IAM users access to the Billing and Cost Management console.](#)
- [Billing Permissions Descriptions](#)

For related information in the *IAM User Guide*, see the following resources:

- [Managed Policies and Inline Policies \(p. 279\)](#)
- [Controlling User Access to the AWS Management Console \(p. 53\)](#)
- [Attaching a Policy to an IAM Group \(p. 132\)](#)

Summary

You've now successfully completed all of the steps necessary to delegate user access to the Billing and Cost Management console. As a result, you've seen firsthand what your users billing console experience will be like and can now proceed to implement this logic in your production environment at your convenience.

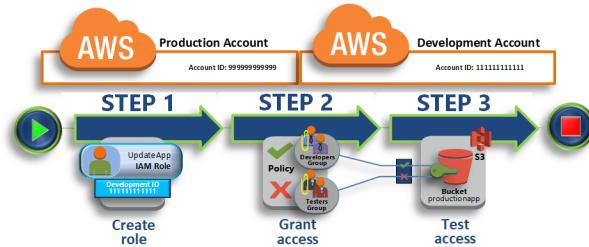
Tutorial: Delegate Access Across AWS Accounts Using IAM Roles

This tutorial teaches you how to use a role to delegate access to resources that are in different AWS accounts that you own (Production and Development). You share resources in one account with users in a different account. By setting up cross-account access in this way, you don't need to create individual IAM users in each account. In addition, users don't have to sign out of one account and sign into another in order to access resources that are in different AWS accounts. After configuring the role, you see how to use the role from the AWS Management Console, the AWS CLI, and the API.

In this tutorial, imagine that the Production account is where live applications are managed, and the Development account is a sandbox where developers and testers can freely test applications. In each account, application information is stored in Amazon S3 buckets. You manage IAM users in the Development account, where you have two IAM groups: Developers and Testers. Users in both groups have permissions to work in the Development account and access resources there. From time to time, a developer must update the live applications in the Production account. These applications are stored in an Amazon S3 bucket called `productionapp`.

At the end of this tutorial, you have a role in the Production account (the trusting account) that allows users from the Development account (the trusted account) to access the `productionapp` bucket in the Production account. Developers can use the role in the AWS Management Console to access the `productionapp` bucket in the Production account. They can also access the bucket by using API calls that are authenticated by temporary credentials provided by the role. Similar attempts by a Tester to use the role fail.

This workflow has three basic steps.



Step 1 - Create a Role (p. 27)

First, you use the AWS Management Console to establish trust between the Production account (ID number 999999999999) and the Development account (ID number 111111111111). You start

by creating an IAM role named *UpdateApp*. When you create the role, you define the Development account as a trusted entity and specify a permissions policy that allows trusted users to update the productionapp bucket.

Step 2 - Grant Access to the Role (p. 29)

In this step of the tutorial, you modify the IAM group policy so that Testers are denied access to the UpdateApp role. Because Testers have PowerUser access in this scenario, we must explicitly deny the ability to use the role.

Step 3 - Test Access by Switching Roles (p. 31)

Finally, as a Developer, you use the UpdateApp role to update the productionapp bucket in the Production account. You see how to access the role through the AWS console, the AWS CLI, and the API.

Prerequisites

This tutorial assumes that you have the following already in place:

- Two separate AWS accounts that you can use, one to represent the Development account, and one to represent the Production account.
- Users and groups in the Development account created and configured as follows:

User	Group	Permissions
David	Developers	Both users are able to sign in and use the AWS Management Console in the Development account.
Theresa	Testers	

- You do not need to have any users or groups created in the Production account.
- An Amazon S3 bucket created in the Production account. We call it `ProductionApp` in this tutorial, but because S3 bucket names must be globally unique, you must use a bucket with a different name.

Step 1 - Create a Role

To allow users from one AWS account to access resources in another AWS account, create a role that defines who can access it and what permissions it grants to users that switch to it.

In this step of the tutorial, you create the role in the Production account and specify the Development account as a trusted entity. You also limit the role's permissions to only read and write access to the productionapp bucket. Anyone who is granted permission to use the role can read and write to the productionapp bucket.

Before you can create a role, you need the account ID of the Development AWS account. The account ID is a unique identifier assigned to each AWS account.

To obtain the Development AWS account ID

1. Sign in to the AWS Management Console as an administrator of the Development account, and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In navigation bar, choose **Support**, and then **Support Center**. The **Account Number** is in the upper right corner immediately below the **Support** menu. The account ID is a 12-digit number. For this scenario, we pretend the Development account ID is 111111111111; however, you should use a valid account ID if you are reconstructing the scenario in your test environment.

To create a role in the Production account that can be used by the Development account

1. Sign in to the AWS Management Console as an administrator of the Production account, and open the IAM console.
2. Before creating the role, prepare the managed policy that defines the permissions that the role requires. You attach this policy to the role in a later step.

You want to set read and write access to the `productionapp` bucket. Although AWS provides some Amazon S3 managed policies, there isn't one that provides read and write access to a single Amazon S3 bucket. You can create your own policy instead.

In the navigation pane on the left, choose **Policies** and then choose **Create policy**.

3. Choose the **JSON** tab and copy the text from the following JSON policy document. Paste this text into the **JSON** text box, replacing the resource ARN (`arn:aws:s3:::productionapp`) with the real one appropriate to your S3 bucket.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "s3>ListAllMyBuckets",  
            "Resource": "arn:aws:s3:::*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3>ListBucket",  
                "s3:GetBucketLocation"  
            ],  
            "Resource": "arn:aws:s3:::productionapp"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3GetObject",  
                "s3PutObject",  
                "s3DeleteObject"  
            ],  
            "Resource": "arn:aws:s3:::productionapp/*"  
        }  
    ]  
}
```

The `ListBucket` permission allows users to view objects in the `productionapp` bucket. The `GetObject`, `PutObject`, `DeleteObject` permissions allow users to view, update, and delete contents in the `productionapp` bucket.

4. When you are finished, choose **Review policy**. The [Policy Validator \(p. 321\)](#) reports any syntax errors.

Note

You can switch between the **Visual editor** and **JSON** tabs any time. However, if you make changes or choose **Review policy** in the **Visual editor** tab, IAM might restructure your policy to optimize it for the visual editor. For more information, see [Policy Restructuring \(p. 387\)](#).

5. On the **Review** page, type `read-write-app-bucket` for the policy name. Review the policy **Summary** to see the permissions granted by your policy, and then choose **Create policy** to save your work.

The new policy appears in the list of managed policies.

6. In the navigation pane on the left, choose **Roles** and then choose **Create role**.

7. Choose the **Another AWS account** role type.
8. For **Account ID**, type the Development account ID.

This tutorial uses the example account ID **111111111111** for the Development account. You should use a valid account ID. If you use an invalid account ID, such as **111111111111**, IAM does not let you create the new role.

For now you do not need to require an external ID, or require users to have multi-factor authentication (MFA) in order to assume the role. So leave these options unselected. For more information, see [Using Multi-Factor Authentication \(MFA\) in AWS \(p. 91\)](#)

9. Choose **Next: Permissions** to set the permissions that will be associated with the role.
10. Select the box next to the policy that you created previously.

Tip

For **Filter**, choose **Customer managed** to filter the list to include only the policies that you have created. This hides the AWS created policies and makes it much easier to find the one you're looking for.

Then choose **Next: Review**.

11. Type **UpdateApp** for the role name.
12. (Optional) For **Role description**, type a description for the new role.
13. After reviewing the role, choose **Create role**.

The **UpdateApp** role appears in the list of roles.

Now you must obtain the role's Amazon Resource Name (ARN), which is a unique identifier for the role. When you modify the Developers and Testers group's policy, you will specify the role's ARN to grant or deny permissions.

To obtain the ARN for UpdateApp

1. In the navigation pane of the IAM console, choose **Roles**.
2. In the list of roles, choose the **UpdateApp** role.
3. In the **Summary** section of the details pane, copy the **Role ARN** value.

The Production account has an account ID of **999999999999**, so the role ARN is **arn:aws:iam::999999999999:role/UpdateApp**. Ensure that you supply the real AWS account ID for your 'production' account.

At this point, you have established trust between the Production and Development accounts by creating a role in the Production account that identifies the Development account as a trusted principal. You also defined what users who switch to the **UpdateApp** role can do.

Next, modify the permissions for the groups.

Step 2 - Grant Access to the Role

At this point, both Testers and Developers group members have permissions that allow them to freely test applications in the Development account. Here are the steps required to add permissions to allow switching to the role.

To modify the Developers group to allow them to switch to the UpdateApp role

1. Sign in as an administrator in the Development account, and open the IAM console.
2. Choose **Groups**, and then choose **Developers**.

3. Choose the **Permissions** tab, expand the **Inline Policies** section, and then choose **Create Group Policy**. If no inline policy exists yet, then the button does not appear. Instead, choose the link at the end of "To create one, click here."
4. Choose **Custom Policy** and then choose **Select** button.
5. Type a policy name like **allow-assume-S3-role-in-production**.
6. Add the following policy statement to allow the AssumeRole action on the UpdateApp role in the Production account. Be sure that you change **PRODUCTION-ACCOUNT-ID** in the Resource element to the actual AWS account ID of the Production account.

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Action": "sts:AssumeRole",  
        "Resource": "arn:aws:iam::PRODUCTION-ACCOUNT-ID:role/UpdateApp"  
    }  
}
```

The Allow effect explicitly allows the Developers group access to the UpdateApp role in the Production account. Any developer who tries to access the role will succeed.

7. Choose **Apply Policy** to add the policy to the Developer group.

In most environments, the following procedure is likely not needed. If, however, you use Power User permissions, then some groups might already be able to switch roles. The following procedures shows how to add a "Deny" permission to the Testers group to ensure that they cannot assume the role. If this procedure is not needed in your environment, then we recommend that you do not add it. "Deny" permissions make the overall permissions picture more complicated to manage and understand. Use "Deny" permissions only when there is not a better option.

To modify the Testers group to deny permission to assume the UpdateApp role

1. Choose **Groups**, and then choose **Testers**.
2. Choose the **Permissions** tab, expand the **Inline Policies** section, and then choose **Create Group Policy**.
3. Choose **Custom Policy** and then choose the **Select** button.
4. Type a policy name like **deny-assume-S3-role-in-production**.
5. Add the following policy statement to deny the AssumeRole action on the UpdateApp role. Be sure that you change **PRODUCTION-ACCOUNT-ID** in the Resource element to the actual AWS account ID of the Production account.

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Deny",  
        "Action": "sts:AssumeRole",  
        "Resource": "arn:aws:iam::PRODUCTION-ACCOUNT-ID:role/UpdateApp"  
    }  
}
```

The Deny effect explicitly denies the Testers group access to the UpdateApp role in the Production account. Any tester who tries to access the role will get an access denied message.

6. Choose **Apply Policy** to add the policy to the Tester group.

The Developers group now has permissions to use the `UpdateApp` role in the Production account. The Testers group is prevented from using the `UpdateApp` role.

Next, you'll learn how David, a developer, can access the `productionapp` bucket in the Production account by using the AWS Management Console, the AWS CLI commands, and the `AssumeRole` API call.

Step 3 - Test Access by Switching Roles

After completing the first two steps of this tutorial, you have a role that grants access to a resource in the Production account. You also have one group in the Development account whose users are allowed to use that role. The role is now ready to use. This step discusses how to test switching to that role from the AWS Management Console, the AWS CLI, and the AWS API.

Important

You can switch to a role only when you are signed in as an IAM user or a federated user.

Additionally, if you launch an Amazon EC2 instance to run an application, the application can assume a role through its instance profile. You cannot switch to a role when you are signed in as the AWS account root user.

Switch Roles (Console)

If David needs to work with in the Production environment in the AWS Management Console, he can do so by using **Switch Role**. He specifies the account ID or alias and the role name, and his permissions immediately switch to those permitted by the role. He can then use the console to work with the `productionapp` bucket, but cannot work with any other resources in Production. While David is using the role, he also cannot make use of his power-user privileges in the Development account. That's because only one set of permissions can be in effect at a time.

Important

Switching roles using the AWS Management Console works only with accounts that do not require an `ExternalID`. If you grant access to your account to a third party and require an `ExternalID` in a `Condition` element in your permission policy, the third party can access your account only by using the AWS API or a command line tool. The third party cannot use the console because it cannot supply a value for `ExternalID`. For more information about this scenario, see [How to Use an External ID When Granting Access to Your AWS Resources to a Third Party \(p. 187\)](#), and [How to Enable Cross-Account Access to the AWS Management Console](#) in the [AWS Security Blog](#).

There are two ways that David can use to enter the **Switch Role** page:

- David receives a link from his administrator that points to a pre-defined Switch Role configuration. The link is provided to the administrator on the final page of the **Create role** wizard or on the **Role Summary** page for a cross-account role. Choosing this link takes David to the **Switch Role** page with the **Account ID** and **Role name** fields already filled in. All David needs to do is choose **Switch Role** and he's done.
- The administrator does not send the link in email, but instead sends the **Account ID** number and **Role Name** values. David must manually type them to switch roles. This is illustrated in the following procedure.

To assume a role

1. David signs into the AWS console using his normal user that is in the Development group.
2. He chooses the link that his administrator sent to him in email. This takes him to the **Switch Role** page with the account ID or alias and the role name information already filled in.

—or—

He chooses his name (the Identity menu) on the navigation bar, and then chooses **Switch Role**.

If this is the first time David tries to access the Switch Role page this way, he will first land on a first-run **Switch Role** page. This page provides additional information on how switching roles can enable users to manage resources across AWS accounts. David must choose the **Switch Role** button on this page to complete the rest of this procedure.

3. Next, in order to access the role, David must manually type the Production account ID number (999999999999) and the role name (UpdateApp).

Also, to help him stay aware of which role (and associated permissions) are currently active, he types **PRODUCTION** in the **Display Name** text box, selects the red color option, and then chooses **Switch Role**.

4. David can now use the Amazon S3 console to work with the Amazon S3 bucket, or any other resource to which the UpdateApp role has permissions.
5. When he is done with the work he needs to do, David can return to his original permissions. To do that, he chooses the **PRODUCTION** role display name on the navigation bar and then chooses **Back to David @ 111111111111**.
6. The next time David wants to switch roles and chooses the Identity menu in the navigation bar, he sees the PRODUCTION entry still there from last time. He can simply choose that entry to switch roles immediately without having to reenter the account ID and role name.

Switch Roles (AWS CLI)

If David needs to work in the Production environment at the command line, he can do so by using the [AWS CLI](#). He runs the `aws sts assume-role` command and passes the role ARN to get temporary security credentials for that role. He then configures those credentials in environment variables so subsequent AWS CLI commands work using the role's permissions. While David is using the role, he cannot use his power-user privileges in the Development account because only one set of permissions can be in effect at a time.

Note that all access keys and tokens are examples only and cannot be used as shown. Replace with the appropriate values from your live environment.

To assume a role

1. David opens a command prompt window, and confirms that the AWS CLI client is working by running the command:

```
aws help
```

Note

David's default environment uses the David user credentials from his default profile that he created with the `aws configure` command. For more information, see [Configuring the AWS Command Line Interface](#) in the [AWS Command Line Interface User Guide](#).

2. He begins the switch role process by running the following command to switch to the UpdateApp role in the Production account. He got the role ARN from the administrator that created the role. The command requires that you provide a session name as well, you can choose any text you like for that.

```
aws sts assume-role --role-arn "arn:aws:iam::999999999999:role/UpdateApp" --role-session-name "David-ProdUpdate"
```

David then sees the following in the output:

```
{
```

```

    "Credentials": {
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "SessionToken": "AQoDYXdzEGcaEXAMPLE2gsYULo
+Im5ZEXAMPEeYjs1M2FUIgIJx9tQqNMBEXAMPLE
CvSRyhOFW7jEXAMPLE+vE/7s1HRpXviG7b+qYf4nD00EXAMPLEmj4wxS04L/
uZEXAMPLEcihzFB5lTYLto9dyBgSDy
EXAMPLE9/
g7QRUhZp4bqbEXAMPLEnGPyOj59pFA41NKCikVgkREXAMPLEj1zxQ7y52gekeVEXAMPLEDiB9ST3Uuysg
sKdEXAMPLE1TVastU1AOSKFEXAMPLEiywCC/Cs8EXAMPLEpZgOs+6hz4AP4KEXAMPLERbASP
+4eZScEXAMPLEsnf87e
NhYDHq6ikBQ==",
        "Expiration": "2014-12-11T23:08:07Z",
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
    }
}

```

3. David sees the three pieces that he needs in the Credentials section of the output.

- AccessKeyId
- SecretAccessKey
- SessionToken

David needs to configure the AWS CLI environment to use these parameters in subsequent calls. For information about the various ways to configure your credentials, see [Configuring the AWS Command Line Interface](#). You cannot use the `aws configure` command because it does not support capturing the session token. However, you can manually type the information into a configuration file. Because these are temporary credentials with a relatively short expiration time, it is easiest to add them to the environment of your current command line session.

4. To add the three values to the environment, David cuts and pastes the output of the previous step into the following commands. Note that you might want to cut and paste into a simple text editor to address line wrap issues in the output of the session token. It must be added as a single long string, even though it is shown line wrapped here for clarity.

Note

The following example shows commands given in the Windows environment, where "set" is the command to create an environment variable. On a Linux or Mac computer, you would use the command "export" instead. All other parts of the example are valid in all three environments.

```

set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
set AWS_SESSION_TOKEN=AQoDYXdzEGcaEXAMPLE2gsYULo
+Im5ZEXAMPEeYjs1M2FUIgIJx9tQqNMBEXAMPLECvS
RyhOFW7jEXAMPLE+vE/7s1HRpXviG7b+qYf4nD00EXAMPLEmj4wxS04L/
uZEXAMPLEcihzFB5lTYLto9dyBgSDyEXA
MLEKEY9/
g7QRUhZp4bqbEXAMPLEnGPyOj59pFA41NKCikVgkREXAMPLEj1zxQ7y52gekeVEXAMPLEDiB9ST3UusKd
EXAMPLE1TVastU1AOSKFEXAMPLEiywCC/Cs8EXAMPLEpZgOs+6hz4AP4KEXAMPLERbASP
+4eZScEXAMPLEnhYkxiHen
DHq6ikBQ==

```

At this point, any following commands run under the permissions of the role identified by those credentials. In David's case, the `UpdateApp` role.

5. Run the command to access the resources in the Production account. In this example, David simply lists the contents of his S3 bucket with the following command.

```
aws s3 ls s3://productionapp
```

Because Amazon S3 bucket names are universally unique, there is no need to specify the account ID that owns the bucket. To access resources for other AWS services, refer to the AWS CLI documentation for that service for the commands and syntax that are required to reference its resources.

Using AssumeRole (AWS API)

When David needs to make an update to the Production account from code, he makes an `AssumeRole` call to assume the `UpdateApp` role. The call returns temporary credentials that he can use to access the `productionapp` bucket in the Production account. With those credentials, David can make API calls to update the `productionapp` bucket. However, he cannot make API calls to access any other resources in the Production account, even though he has power-user permissions in the Development account.

To assume a role

1. David calls `AssumeRole` as part of an application. He must specify the `UpdateApp` ARN:
`arn:aws:iam::999999999999:role/UpdateApp`.

The response from the `AssumeRole` call includes the temporary credentials with an `AccessKeyId`, a `SecretAccessKey`, and an `Expiration` time that indicates when the credentials expire and you must request new ones.

2. With the temporary credentials, David makes an `s3:PutObject` call to update the `productionapp` bucket. He would pass the credentials to the API call as the `AuthParams` parameter. Because the temporary role credentials have only read and write access to the `productionapp` bucket, any other actions in the Production account are denied.

For a code example (using Python), see [Switching to an IAM Role \(API\) \(p. 214\)](#).

Related Resources

- For more information about IAM users and groups, see [Identities \(Users, Groups, and Roles\) \(p. 59\)](#).
- For more information about Amazon S3 buckets, see [Create a Bucket](#) in the *Amazon Simple Storage Service Getting Started Guide*.

Summary

You have completed the cross-account API access tutorial. You created a role to establish trust with another account and defined what actions trusted entities can take. Then, you modified a group policy to control which IAM users can access the role. As a result, developers from the Development account can make updates to the `productionapp` bucket in the Production account by using temporary credentials.

Tutorial: Create and Attach Your First Customer Managed Policy

In this tutorial, you use the AWS Management Console to create a [customer managed policy \(p. 281\)](#) and then attach that policy to an IAM user in your AWS account. The policy you create allows an IAM test user to sign in directly to the AWS Management Console with read-only permissions.

This workflow has three basic steps:

Step 1: Create the Policy (p. 35)

By default, IAM users do not have permissions to do anything. They cannot access the AWS Management Console or manage the data within unless you allow it. In this step, you create a customer managed policy that allows any attached user to sign in to the console.

Step 2: Attach the Policy (p. 36)

When you attach a policy to a user, the user inherits all of the access permissions that are associated with that policy. In this step, you attach the new policy to a test user account.

Step 3: Test User Access (p. 36)

Once the policy is attached, you can sign in as the user and test the policy.

Prerequisites

To perform the steps in this tutorial, you need to already have the following:

- An AWS account that you can sign in to as an IAM user with administrative permissions.
- A test IAM user that has no permissions assigned or group memberships as follows:

User Name	Group	Permissions
PolicyUser	<none>	<none>

Step 1: Create the Policy

In this step, you create a customer managed policy that allows any attached user to sign in to the AWS Management Console with read-only access to IAM data.

To create the policy for your test user

1. Sign in to the IAM console at <https://console.aws.amazon.com/iam/> with your user that has administrator permissions.
2. In the navigation pane, choose **Policies**.
3. In the content pane, choose **Create policy**.
4. Choose the **JSON** tab and copy the text from the following JSON policy document. Paste this text into the **JSON** text box.

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

5. When you are finished, choose **Review policy**. The [Policy Validator \(p. 321\)](#) reports any syntax errors.

Note

You can switch between the **Visual editor** and **JSON** tabs any time. However, if you make changes or choose **Review policy** in the **Visual editor** tab, IAM might restructure your policy to optimize it for the visual editor. For more information, see [Policy Restructuring \(p. 387\)](#).

6. On the **Review** page, type **UsersReadOnlyAccessToIAMConsole** for the policy name. Review the policy **Summary** to see the permissions granted by your policy, and then choose **Create policy** to save your work.

The new policy appears in the list of managed policies and is ready to attach.

Step 2: Attach the Policy

Next you attach the policy you just created to your test IAM user.

To attach the policy to your test user

1. In the IAM console, in the navigation pane, choose **Policies**.
2. At the top of the policy list, in the search box, start typing **UsersReadOnlyAccessToIAMConsole** until you can see your policy, and then check the box next to **UsersReadOnlyAccessToIAMConsole** in the list.
3. Choose the **Policy actions** button, and then chose **Attach**.
4. For **Filter**, choose **Users**.
5. In the search box, start typing **PolicyUser** until that user is visible on the list, and then check the box next to that user in the list.
6. Choose **Attach Policy**.

You have attached the policy to your IAM test user, which means that user now has read-only access to the IAM console.

Step 3: Test User Access

For this tutorial, we recommend that you test access by signing in as the test user so you can observe the results and see what your users might experience.

To test access by signing in with your test user account

1. Sign in to the IAM console at <https://console.aws.amazon.com/iam/> with your **PolicyUser** test user.
2. Browse through the pages of the console and try to create a new user or group. Notice that **PolicyUser** can display data but cannot create or modify existing IAM data.

Related Resources

For related information in the *IAM User Guide*, see the following resources:

- [Managed Policies and Inline Policies \(p. 279\)](#)
- [Controlling User Access to the AWS Management Console \(p. 53\)](#)
- [Create Individual IAM Users \(p. 44\)](#)

Summary

You've now successfully completed all of the steps necessary to create and attach a customer managed policy. As a result, you are able to sign in to the IAM console with your test account and have seen firsthand what the experience would be like for your users.

Tutorial: Enable Your Users to Configure Their Own Credentials and MFA Settings

You can enable your users to self-manage their own multi-factor authentication (MFA) devices and credentials. You can use the AWS Management Console to configure credentials (access keys, passwords, signing certificates, and SSH public keys) and MFA devices for your users in small numbers. But that is a task that could very quickly become time consuming as the number of users grows. Security best practice specifies that users should regularly change their passwords and rotate their access keys. They should also delete or deactivate credentials that are not needed and use MFA, at the very least, for sensitive operations. Showing you how to enable these best practices without burdening your administrators is the goal of this tutorial.

This tutorial shows how to grant users access to AWS services, but **only** when they sign in with MFA. If they are not signed in with an MFA device, then users cannot access other services.

This workflow has three basic steps.

Step 1: Create a Policy to Enforce MFA Sign-In (p. 38)

Create a customer managed policy that prohibits all actions **except** the few IAM API operations that enable changing credentials and managing MFA devices.

Step 2: Attach Policies to Your Test Group (p. 41)

Create a group whose members have full access to all Amazon EC2 actions if they sign-in with MFA. To create such a group, you attach both the AWS managed policy called `AmazonEC2FullAccess` and the customer managed policy you created in the first step.

Step 3: Test Your User's Access (p. 41)

Sign in as the test user to verify that access to Amazon EC2 is blocked *until* the user creates an MFA device and then signs in using that device.

Prerequisites

To perform the steps in this tutorial, you must already have the following:

- An AWS account that you can sign in to as an IAM user with administrative permissions.
- Your account ID number which you type into the policy in Step 1.

To find your account ID number, on the navigation bar at the top of the page, choose **Support** and then choose **Support Center**. You can find your account ID under this page's **Support** menu.

- A test IAM user who is a member of a group as follows:

Create user account		Create and configure group account		
MFAUser	Choose only the option for AWS Management Console access , and assign a password.	EC2MFA	MFAUser	Do NOT attach any policies or otherwise grant permissions to this group.

Step 1: Create a Policy to Enforce MFA Sign-In

You begin by creating an IAM customer managed policy that denies all permissions except those required for IAM users to manage their own credentials and MFA devices.

1. Sign in to the AWS Management Console as a user with administrator credentials. To adhere to IAM best practices, don't sign in with your AWS account root user credentials. For more information, see [Create individual IAM users](#).
2. Open the IAM console at <https://console.aws.amazon.com/iam/>.
3. In the navigation pane, choose **Policies**, and then choose **Create policy**.
4. Choose the **JSON** tab and copy the text from the following JSON policy document. Paste this text into the **JSON** text box.

Note

This example policy does not allow users to both sign-in and perform a password change. New users and users with an expired password might try to do so. To allow this, move `iam:ChangePassword` and `iam:CreateLoginProfile` from the statement `AllowIndividualUserToSeeAndManageOnlyTheirOwnAccountInformation` to the statement `BlockMostAccessUnlessSignedInWithMFA`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllUsersToListAccounts",
            "Effect": "Allow",
            "Action": [
                "iam>ListAccountAliases",
                "iam>ListUsers",
                "iam>ListVirtualMFADevices",
                "iam>GetAccountPasswordPolicy",
                "iam>GetAccountSummary"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowIndividualUserToSeeAndManageOnlyTheirOwnAccountInformation",
            "Effect": "Allow",
            "Action": [
                "iam>ChangePassword",
                "iam>CreateAccessKey",
                "iam>CreateLoginProfile",
                "iam>DeleteAccessKey",
                "iam>DeleteLoginProfile",
                "iam>GetLoginProfile",
                "iam>ListAccessKeys",
                "iam>UpdateAccessKey",
                "iam>UpdateLoginProfile",
                "iam>ListSigningCertificates",
                "iam>DeleteSigningCertificate",
                "iam>UpdateSigningCertificate",
                "iam>GetAccountSummary"
            ],
            "Resource": "*"
        }
    ]
}
```

```
        "iam:UploadSigningCertificate",
        "iam>ListSSHPublicKeys",
        "iam:GetSSHPublicKey",
        "iam>DeleteSSHPublicKey",
        "iam:UpdateSSHPublicKey",
        "iam:UploadSSHPublicKey"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
},
{
    "Sid": "AllowIndividualUserToListOnlyTheirOwnMFA",
    "Effect": "Allow",
    "Action": [
        "iam>ListMFADevices"
    ],
    "Resource": [
        "arn:aws:iam::*:mfa/*",
        "arn:aws:iam::*:user/${aws:username}"
    ]
},
{
    "Sid": "AllowIndividualUserToManageTheirOwnMFA",
    "Effect": "Allow",
    "Action": [
        "iam>CreateVirtualMFADevice",
        "iam>DeleteVirtualMFADevice",
        "iam:EnableMFADevice",
        "iam:ResyncMFADevice"
    ],
    "Resource": [
        "arn:aws:iam::*:mfa/${aws:username}",
        "arn:aws:iam::*:user/${aws:username}"
    ]
},
{
    "Sid": "AllowIndividualUserToDeactivateOnlyTheirOwnMFAOnlyWhenUsingMFA",
    "Effect": "Allow",
    "Action": [
        "iam:DeactivateMFADevice"
    ],
    "Resource": [
        "arn:aws:iam::*:mfa/${aws:username}",
        "arn:aws:iam::*:user/${aws:username}"
    ],
    "Condition": {
        "Bool": {
            "aws:MultiFactorAuthPresent": "true"
        }
    }
},
{
    "Sid": "BlockMostAccessUnlessSignedInWithMFA",
    "Effect": "Deny",
    "NotAction": [
        "iam>CreateVirtualMFADevice",
        "iam>DeleteVirtualMFADevice",
        "iam>ListVirtualMFADevices",
        "iam:EnableMFADevice",
        "iam:ResyncMFADevice",
        "iam>ListAccountAliases",
        "iam>ListUsers",
        "iam>ListSSHPublicKeys",
        "iam>ListAccessKeys",
        "iam>ListServiceSpecificCredentials",
        "iam>ListMFADevices",
        "iam:GetAccountSummary",
    ]
}
```

```
        "sts:GetSessionToken"
    ],
    "Resource": "*",
    "Condition": {
        "BoolIfExists": {
            "aws:MultiFactorAuthPresent": "false"
        }
    }
}
```

What does this policy do?

- The first statement enables the user to see basic information about the account and its users in the IAM console. These permissions must be in their own statement because they do not support or do not need to specify a specific resource ARN, and instead specify "Resource" : "*".
 - The second statement enables the user to manage his or her own user, password, access keys, signing certificates, SSH public keys, and MFA information in the IAM console. The resource ARN limits the use of these permissions to only the user's own IAM user entity.
 - The third statement enables the user to see information about MFA devices, and which are associated with his or her IAM user entity.
 - The fourth statement allows the user to provision or manage his or her own MFA device. Notice that the resource ARNs in the fourth statement allow access to only an MFA device or user that has the exact same name as the currently signed-in user. Users can't create or alter any MFA device other than their own.
 - The fifth statement allows the user to deactivate only his or her own MFA device and only if the user signed in using MFA. This prevents others with only the access keys (and not the MFA device) from deactivating the MFA device and replacing it with their own.
 - The sixth and final statement uses a combination of "Deny" and "NotAction" to deny all actions for all other AWS services *if* the user is not signed-in with MFA. If the user is signed-in with MFA, then the "Condition" test fails and the final "deny" statement has no effect and other permissions granted to the user can take effect. This last statement ensures that when the user is not signed-in with MFA that they can perform only the IAM actions allowed in the earlier statements. The ...IfExists version of the Bool operator ensures that if the aws:MultiFactorAuthPresent key is missing, the condition returns true. This means that a user accessing an API with long-term credentials, such as an access key, is denied access to the non-IAM API operations.
5. When you are finished, choose **Review policy**. The [Policy Validator \(p. 321\)](#) reports any syntax errors.

Note

You can switch between the **Visual editor** and **JSON** tabs any time. However, the policy above includes the `NotAction` element, which is not supported in the visual editor. For this policy, you will see a notification on the **Visual editor** tab. Return to the **JSON** tab to continue working with this policy.

6. On the **Review** page, type **Force_MFA** for the policy name. For the policy description, type **This policy allows users to manage their own passwords and MFA devices but nothing else unless they authenticate with MFA.** Review the policy **Summary** to see the permissions granted by your policy, and then choose **Create policy** to save your work.

The new policy appears in the list of managed policies and is ready to attach.

Step 2: Attach Policies to Your Test Group

Next you attach two policies to the test IAM group, which will be used to grant the MFA-protected permissions.

1. In the navigation pane, choose **Groups**.
2. In the search box, type **EC2MFA**, and then select the group name (not the check box) in the list.
3. On the **Permissions** tab, and click **Attach Policy**.
4. On the **Attach Policy** page, in the search box, type **EC2Full** and then select the check box next to **AmazonEC2FullAccess** in the list. Don't save your changes yet.
5. In the search box, type **Force**, and then select the check box next to **Force_MFA** in the list.
6. Choose **Attach Policy**.

Step 3: Test Your User's Access

In this part of the tutorial, you sign in as the test user and verify that the policy works as intended.

1. Sign in to your AWS account as **MFAUser** with the password you assigned in the previous section. Use the URL: <https://<alias or account ID number>.signin.aws.amazon.com/console>
2. Choose **EC2** to open the Amazon EC2 console and verify that the user has no permissions to do anything.
3. On the navigation bar, choose **Services**, and then choose **IAM** to open the IAM console.
4. In the navigation pane, choose **Users**, and then choose the user (not the check box) **MFAUser**. If the **Groups** tab appears by default, note that it says that you don't have permissions to see your group memberships.
5. Now add an MFA device. Choose the **Security credentials** tab. Next to **Assigned MFA device**, choose the edit icon (✎).
6. For this tutorial, we use a virtual (software-based) MFA device, such as the Google Authenticator app on a mobile phone. Choose **A virtual MFA device**, and then click **Next Step**.

IAM generates and displays configuration information for the virtual MFA device, including a QR code graphic. The graphic is a representation of the secret configuration key that is available for manual entry on devices that do not support QR codes.

7. Open your virtual MFA app. (For a list of apps that you can use for hosting virtual MFA devices, see [Virtual MFA Applications](#).) If the virtual MFA app supports multiple accounts (multiple virtual MFA devices), choose the option to create a new account (a new virtual MFA device).
8. Determine whether the MFA app supports QR codes, and then do one of the following:
 - Use the app to scan the QR code. For example, you might choose the camera icon or choose an option similar to **Scan code**, and then use the device's camera to scan the code.
 - In the **Manage MFA Device** wizard, choose **Show secret key for manual configuration**, and then type the secret configuration key into your MFA app.

When you are finished, the virtual MFA device starts generating one-time passwords.

9. In the **Manage MFA Device** wizard, in the **Authentication Code 1** box, type the one-time password that currently appears in the virtual MFA device. Wait up to 30 seconds for the device to generate a new one-time password. Then type the second one-time password into the **Authentication Code 2** box. Choose **Active Virtual MFA**.

Important

Submit your request immediately after generating the codes. If you generate the codes and then wait too long to submit the request, the MFA device is successfully associated with the user. However, if the MFA device is out of sync. This happens because time-based one-time passwords (TOTP) expire after a short period of time. If this happens, you can [resync the device \(p. 102\)](#).

The virtual MFA device is now ready to use with AWS.

10. Sign out of the console and then sign in as **MFAUser** again. This time AWS prompts you for an MFA code from your phone. When you get it, type the code in the box and then choose **Submit**.
11. Choose **EC2** to open the Amazon EC2 console again. Note that this time you can see all the information and perform any actions you wish. If you go to any other console as this user, you see access denied messages because the policies in this tutorial grant access only to Amazon EC2.

Related Resources

For related information found in the *IAM User Guide*, see the following resources:

- [Using Multi-Factor Authentication \(MFA\) in AWS \(p. 91\)](#)
- [Enabling MFA Devices \(p. 92\)](#)
- [Using MFA Devices With Your IAM Sign-in Page \(p. 56\)](#)

IAM Best Practices and Use Cases

To get the greatest benefits from IAM, take time to learn the recommended best practices. One way to do this is to see how IAM is used in real-world scenarios to work with other AWS services.

Topics

- [IAM Best Practices \(p. 43\)](#)
- [Business Use Cases \(p. 49\)](#)

IAM Best Practices



To help secure your AWS resources, follow these recommendations for the AWS Identity and Access Management (IAM) service.

Topics

- [Lock Away Your AWS Account Root User Access Keys \(p. 43\)](#)
- [Create Individual IAM Users \(p. 44\)](#)
- [Use AWS Defined Policies to Assign Permissions Whenever Possible \(p. 44\)](#)
- [Use Groups to Assign Permissions to IAM Users \(p. 44\)](#)
- [Grant Least Privilege \(p. 45\)](#)
- [Use Access Levels to Review IAM Permissions \(p. 45\)](#)
- [Configure a Strong Password Policy for Your Users \(p. 46\)](#)
- [Enable MFA for Privileged Users \(p. 46\)](#)
- [Use Roles for Applications That Run on Amazon EC2 Instances \(p. 46\)](#)
- [Delegate by Using Roles Instead of by Sharing Credentials \(p. 47\)](#)
- [Rotate Credentials Regularly \(p. 47\)](#)
- [Remove Unnecessary Credentials \(p. 47\)](#)
- [Use Policy Conditions for Extra Security \(p. 47\)](#)
- [Monitor Activity in Your AWS Account \(p. 48\)](#)
- [Video Presentation About IAM Best Practices \(p. 48\)](#)

Lock Away Your AWS Account Root User Access Keys

You use an access key (an access key ID and secret access key) to make programmatic requests to AWS. However, do not use your AWS account root user access key. The access key for your AWS account gives full access to all your resources for all AWS services, including your billing information. You cannot restrict the permissions associated with your AWS account access key.

Therefore, protect your AWS account access key like you would your credit card numbers or any other sensitive secret. Here are some ways to do that:

- If you don't already have an access key for your AWS account, don't create one unless you absolutely need to. Instead, use your account email address and password to sign in to the [AWS Management Console](#) and create an IAM user for yourself that has administrative privileges, as explained in the next section.
- If you do have an access key for your AWS account, delete it. If you must keep it, rotate (change) the access key regularly. To delete or rotate your AWS account access keys, go to the [Security Credentials page](#) in the AWS Management Console and sign in with your account's email address and password. You can manage your access keys in the [Access keys](#) section.
- Never share your AWS account password or access keys with anyone. The remaining sections of this document discuss various ways to avoid having to share your AWS account root user credentials with other users and to avoid having to embed them in an application.
- Use a strong password to help protect account-level access to the AWS Management Console. For information about managing your AWS account root user password, see [Changing the AWS Account Root User Password \(p. 75\)](#).
- Enable AWS multi-factor authentication (MFA) on your AWS account. For more information, see [Using Multi-Factor Authentication \(MFA\) in AWS \(p. 91\)](#).

Create Individual IAM Users

Don't use your AWS account root user credentials to access AWS, and don't give your credentials to anyone else. Instead, create individual users for anyone who needs access to your AWS account. Create an IAM user for yourself as well, give that user administrative privileges, and use that IAM user for all your work. For information about how to do this, see [Creating Your First IAM Admin User and Group \(p. 17\)](#).

By creating individual IAM users for people accessing your account, you can give each IAM user a unique set of security credentials. You can also grant different permissions to each IAM user. If necessary, you can change or revoke an IAM user's permissions any time. (If you give out your root user credentials, it can be difficult to revoke them, and it is impossible to restrict their permissions.)

Note

Before you set permissions for individual IAM users, though, see the next point about groups.

Use AWS Defined Policies to Assign Permissions Whenever Possible

We recommend that you use the managed policies that are created and maintained by AWS to grant permissions whenever possible. A key advantage of using these policies is that they are maintained and updated by AWS as new services or new APIs are introduced.

AWS-managed policies are designed to support common tasks. They typically provide access to a single service or a limited set of actions. For more information about AWS managed policies, see [AWS Managed Policies \(p. 280\)](#).

AWS managed policies for job functions can span multiple services and align with common job functions in the IT industry. For a list and descriptions of job function policies, see [AWS Managed Policies for Job Functions \(p. 468\)](#).

Use Groups to Assign Permissions to IAM Users

Instead of defining permissions for individual IAM users, it's usually more convenient to create groups that relate to job functions (administrators, developers, accounting, etc.). Next, define the relevant permissions for each group. Finally, assign IAM users to those groups. All the users in an IAM group

inherit the permissions assigned to the group. That way, you can make changes for everyone in a group in just one place. As people move around in your company, you can simply change what IAM group their IAM user belongs to.

For more information, see the following:

- [Creating Your First IAM Admin User and Group \(p. 17\)](#)
- [Managing IAM Groups \(p. 131\)](#)

Grant Least Privilege

When you create IAM policies, follow the standard security advice of granting *least privilege*—that is, granting only the permissions required to perform a task. Determine what users need to do and then craft policies for them that let the users perform *only* those tasks.

Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later.

Defining the right set of permissions requires some research. Determine what is required for the specific task, what actions a particular service supports, and what permissions are required in order to perform those actions. To have access levels help you determine what permissions are required, see [Use Access Levels to Review IAM Permissions \(p. 45\)](#).

One feature that can help with this is the **Access Advisor** tab. This tab is available on the IAM console details page whenever you inspect a user, group, role, or policy. The tab includes information about which services are actually used by a user, group, role, or by anyone that uses a policy. You can use this information to identify unnecessary permissions so that you can refine your IAM policies to better adhere to the principle of least privilege. For more information, see [Reducing Policy Scope by Viewing User Activity \(p. 342\)](#).

For more information, see the following:

- [Access Management \(p. 274\)](#)
- Policy topics for individual services, which provide examples of how to write policies for service-specific resources. Examples:
 - [Authentication and Access Control for Amazon DynamoDB](#) in the *Amazon DynamoDB Developer Guide*
 - [Using Bucket Policies and User Policies](#) in the *Amazon Simple Storage Service Developer Guide*
 - [Access Control List \(ACL\) Overview](#) in the *Amazon Simple Storage Service Developer Guide*

Use Access Levels to Review IAM Permissions

To improve the security of your AWS account, you should regularly review and monitor each of your IAM policies. Make sure that your policies grant the [least privilege \(p. 45\)](#) that is needed to perform only the necessary actions.

When you review a policy, you can view the [policy summary \(p. 347\)](#) that includes a summary of the access level for each service within that policy. AWS categorizes each service action into one of four *access levels* based on what each action does: List, Read, Write, or Permissions management. You can use these access levels to determine which actions to include in your policies.

For example, in the Amazon S3 service, you might want to allow a large group of users to access List and Read actions. Such actions permit those users to list the buckets and get objects in Amazon S3. However, you should allow only a small group of users to access the Amazon S3 Write actions to delete buckets or put objects into an S3 bucket. Additionally, you should restrict permissions to allow only

administrators to access the Amazon S3 Permissions management actions. This ensures that only a limited number of people can manage bucket policies in Amazon S3. This is especially important for Permissions management actions in IAM and AWS Organizations services.

Important

Before you create or edit policies based on these access levels, review the definitions for each access level. Make sure that the actions you want are categorized the way you expect. To learn about each AWS access level and to view a list of actions that belong to each action level for a specific service, see [IAM Policy Actions Grouped by Access Level \(p. 615\)](#).

To see the access levels for a policy, you must first locate the policy's summary. The policy summary is included on the **Policies** page for managed policies, and on the **Users** page for policies that are attached to a user. For more information, see [Policy Summary \(List of Services\) \(p. 348\)](#).

Within a policy summary, the **Access level** column shows that the policy provides **Full** or **Limited** access to one or more of the four AWS access levels for the service. Alternately, it might show that the policy provides **Full access** to all the actions within the service. You can use the information within this **Access level** column to understand the level of access that the policy provides. You can then take action to make your AWS account more secure. For details and examples of the access level summary, see [Understanding Access Level Summaries Within Policy Summaries \(p. 357\)](#).

Configure a Strong Password Policy for Your Users

If you allow users to change their own passwords, require that they create strong passwords and that they rotate their passwords periodically. On the [Account Settings](#) page of the IAM console, you can create a password policy for your account. You can use the password policy to define password requirements, such as minimum length, whether it requires non-alphabetic characters, how frequently it must be rotated, and so on.

For more information, see [Setting an Account Password Policy for IAM Users \(p. 75\)](#).

Enable MFA for Privileged Users

For extra security, enable multi-factor authentication (MFA) for privileged IAM users (users who are allowed access to sensitive resources or APIs). With MFA, users have a device that generates a unique authentication code (a one-time password, or OTP). Users must provide both their normal credentials (like their user name and password) and the OTP. The MFA device can either be a special piece of hardware, or it can be a virtual device (for example, it can run in an app on a smartphone).

For more information, see [Using Multi-Factor Authentication \(MFA\) in AWS \(p. 91\)](#).

Use Roles for Applications That Run on Amazon EC2 Instances

Applications that run on an Amazon EC2 instance need credentials in order to access other AWS services. To provide credentials to the application in a secure way, use **IAM roles**. A role is an entity that has its own set of permissions, but that isn't a user or group. Roles also don't have their own permanent set of credentials the way IAM users do. In the case of Amazon EC2, IAM dynamically provides temporary credentials to the EC2 instance, and these credentials are automatically rotated for you.

When you launch an EC2 instance, you can specify a role for the instance as a launch parameter. Applications that run on the EC2 instance can use the role's credentials when they access AWS resources. The role's permissions determine what the application is allowed to do.

For more information, see [Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances \(p. 215\)](#).

Delegate by Using Roles Instead of by Sharing Credentials

You might need to allow users from another AWS account to access resources in your AWS account. If so, don't share security credentials, such as access keys, between accounts. Instead, use IAM roles. You can define a role that specifies what permissions the IAM users in the other account are allowed. You can also designate which AWS accounts have the IAM users that are allowed to assume the role.

For more information, see [Roles Terms and Concepts \(p. 135\)](#).

Rotate Credentials Regularly

Change your own passwords and access keys regularly, and make sure that all IAM users in your account do as well. That way, if a password or access key is compromised without your knowledge, you limit how long the credentials can be used to access your resources. You can apply a password policy to your account to require all your IAM users to rotate their passwords, and you can choose how often they must do so.

For more information about setting a password policy in your account, see [Setting an Account Password Policy for IAM Users \(p. 75\)](#).

For more information about rotating access keys for IAM users, see [Rotating Access Keys \(p. 88\)](#).

Remove Unnecessary Credentials

Remove IAM user credentials (that is, passwords and access keys) that are not needed. For example, an IAM user that is used for an application does not need a password (passwords are necessary only to sign in to AWS websites). Similarly, if a user does not and will never use access keys, there's no reason for the user to have them. Passwords and access keys that have not been used recently might be good candidates for removal. You can find unused passwords or access keys using the console, using the API, or by downloading the credentials report.

For more information about finding IAM user credentials that have not been used recently, see [Finding Unused Credentials \(p. 117\)](#).

For more information about deleting passwords for an IAM user, see [Managing Passwords for IAM Users \(p. 79\)](#).

For more information about deactivating or deleting access keys for an IAM user, see [Managing Access Keys for IAM Users \(p. 85\)](#).

For more information about IAM credential reports, see [Getting Credential Reports for Your AWS Account \(p. 120\)](#).

Use Policy Conditions for Extra Security

To the extent that it's practical, define the conditions under which your IAM policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also specify that a request is allowed only within a specified date range or time range. You can also set conditions that require the use of SSL or MFA (multi-factor authentication). For example, you can require that a user has authenticated with an MFA device in order to be allowed to terminate an Amazon EC2 instance.

For more information, see [IAM JSON Policy Elements: Condition \(p. 438\)](#) in the IAM Policy Elements Reference.

Monitor Activity in Your AWS Account

You can use logging features in AWS to determine the actions users have taken in your account and the resources that were used. The log files show the time and date of actions, the source IP for an action, which actions failed due to inadequate permissions, and more.

Logging features are available in the following AWS services:

- [Amazon CloudFront](#) – Logs user requests that CloudFront receives. For more information, see [Access Logs](#) in the *Amazon CloudFront Developer Guide*.
- [AWS CloudTrail](#) – Logs AWS API calls and related events made by or on behalf of an AWS account. For more information, see the [AWS CloudTrail User Guide](#).
- [Amazon CloudWatch](#) – Monitors your AWS Cloud resources and the applications you run on AWS. You can set alarms in CloudWatch based on metrics that you define. For more information, see the [Amazon CloudWatch User Guide](#).
- [AWS Config](#) – Provides detailed historical information about the configuration of your AWS resources, including your IAM users, groups, roles, and policies. For example, you can use AWS Config to determine the permissions that belonged to a user or group at a specific time. For more information, see the [AWS Config Developer Guide](#).
- [Amazon Simple Storage Service \(Amazon S3\)](#) – Logs access requests to your Amazon S3 buckets. For more information, see [Server Access Logging](#) in the *Amazon Simple Storage Service Developer Guide*.

Video Presentation About IAM Best Practices

The following video includes a conference presentation that covers these best practices and shows additional details about how to work with the features discussed here.

AWS re:Invent 2015 - IAM Best Practices

Business Use Cases

A simple business use case for IAM can help you understand basic ways you might implement the service to control the AWS access that your users have. The use case is described in general terms, without the mechanics of how you'd use the IAM API to achieve the results you want.

This use case looks at two typical ways a fictional company called Example Corp might use IAM. The first scenario considers Amazon Elastic Compute Cloud (Amazon EC2). The second considers Amazon Simple Storage Service (Amazon S3).

For more information about using IAM with other services from AWS, see [AWS Services That Work with IAM \(p. 417\)](#).

Topics

- [Initial Setup of Example Corp \(p. 49\)](#)
- [Use Case for IAM with Amazon EC2 \(p. 49\)](#)
- [Use Case for IAM with Amazon S3 \(p. 50\)](#)

Initial Setup of Example Corp

Joe is the founder of Example Corp. Upon starting the company, he creates his own AWS account and uses AWS products by himself. Then he hires employees to work as developers, admins, testers, managers, and system administrators.

Joe uses the AWS Management Console with the AWS account root user credentials to create a user for himself called *Joe*, and a group called *Admins*. He gives the *Admins* group permissions to perform all actions on all the AWS account's resources using the AWS managed policy [AdministratorAccess](#). Then he adds the *Joe* user to the *Admins* group. For a step-by-step guide to creating an Administrators group and an IAM user for yourself, then adding your user to the Administrators group, see [Creating Your First IAM Admin User and Group \(p. 17\)](#).

At this point, Joe can stop using the root user's credentials to interact with AWS, and instead he begins using only his user credentials.

Joe also creates a group called *AllUsers* so that he can easily apply any account-wide permissions to all users in the AWS account. He adds himself to the group. He then creates a group called *Developers*, a group called *Testers*, a group called *Managers*, and a group called *SysAdmins*. He creates users for each of his employees, and puts the users in their respective groups. He also adds them all to the *AllUsers* group. For information about creating groups, see [Creating IAM Groups \(p. 131\)](#). For information about creating users, see [Creating an IAM User in Your AWS Account \(p. 63\)](#). For information about adding users to groups, see [Managing IAM Groups \(p. 131\)](#).

Use Case for IAM with Amazon EC2

A company like Example Corp typically uses IAM to interact with services like Amazon EC2. To understand this part of the use case, you need a basic understanding of Amazon EC2. For more information about Amazon EC2, go to the [Amazon EC2 User Guide for Linux Instances](#).

Amazon EC2 Permissions for the Groups

To provide "perimeter" control, Joe attaches a policy to the *AllUsers* group. This policy denies any AWS request from a user if the originating IP address is outside Example Corp's corporate network.

At Example Corp, different groups require different permissions:

- **System Administrators** – Need permission to create and manage AMIs, instances, snapshots, volumes, security groups, and so on. Joe attaches a policy to the SysAdmins group that gives members of the group permission to use all the Amazon EC2 actions.
- **Developers** – Need the ability to work with instances only. Joe therefore attaches a policy to the Developers group that allows developers to call `DescribeInstances`, `RunInstances`, `StopInstances`, `StartInstances`, and `TerminateInstances`.

Note

Amazon EC2 uses SSH keys, Windows passwords, and security groups to control who has access to the operating system of specific Amazon EC2 instances. There's no method in the IAM system to allow or deny access to the operating system of a specific instance.

- **Managers** – Should not be able to perform any Amazon EC2 actions except listing the Amazon EC2 resources currently available. Therefore, Joe attaches a policy to the Managers group that only lets them call Amazon EC2 "Describe" APIs.

For examples of what these policies might look like, see [Example Policies \(p. 295\)](#) and [Using AWS Identity and Access Management](#) in the *Amazon EC2 User Guide for Linux Instances*.

User's Role Change

At some point, one of the developers, Don, changes roles and becomes a manager. Joe moves Don from the Developers group to the Managers group. Now that he's in the Managers group, Don's ability to interact with Amazon EC2 instances is limited. He can't launch or start instances. He also can't stop or terminate existing instances, even if he was the user who launched or started the instance. He can list only the instances that Example Corp users have launched.

Use Case for IAM with Amazon S3

Companies like Example Corp would also typically use IAM with Amazon S3. Joe has created an Amazon S3 bucket for the company called *example_bucket*.

Creation of Other Users and Groups

As employees, Don and Mary each need to be able to create their own data in the company's bucket. They also need to read and write shared data that all developers work on. To enable this, Joe logically arranges the data in *example_bucket* using an Amazon S3 key prefix scheme as shown in the following figure.

```
/example_bucket
  /home
    /don
    /mary
  /share
    /developers
    /managers
```

Joe divides the master */example_bucket* into a set of home directories for each employee, and a shared area for groups of developers and managers.

Now Joe creates a set of policies to assign permissions to the users and groups:

- **Home directory access for Don** – Joe attaches a policy to Don that lets him read, write, and list any objects with the Amazon S3 key prefix */example_bucket/home/don/*

- **Home directory access for Mary** – Joe attaches a policy to Mary that lets her read, write, and list any objects with the Amazon S3 key prefix `/example_bucket/home/mary/`
- **Shared directory access for the Developers group** – Joe attaches a policy to the group that lets developers read, write, and list any objects in `/example_bucket/share/developers/`
- **Shared directory access for the Managers group** – Joe attaches a policy to the group that lets managers read, write, and list objects in `/example_bucket/share/managers/`

Note

Amazon S3 doesn't automatically give a user who creates a bucket or object permission to perform other actions on that bucket or object. Therefore, in your IAM policies, you must explicitly give users permission to use the Amazon S3 resources they create.

For examples of what these policies might look like, see [Access Control](#) in the *Amazon Simple Storage Service Developer Guide*. For information on how policies are evaluated at run time, see [IAM JSON Policy Evaluation Logic \(p. 458\)](#).

User's Role Change

At some point, one of the developers, Don, changes roles and becomes a manager. We assume that he no longer needs access to the documents in the `share/developers` directory. Joe, as an admin, moves Don to the `Managers` group and out of the `Developers` group. With just that simple reassignment, Don automatically gets all permissions granted to the `Managers` group, but can no longer access data in the `share/developers` directory.

Integration with a Third-Party Business

Organizations often work with partner companies, consultants, and contractors. Example Corp has a partner called the Widget Company, and a Widget Company employee named Natalie needs to put data into a bucket for Example Corp's use. Joe creates a group called `WidgetCo` and a user named Natalie and adds Natalie to the `WidgetCo` group. Joe also creates a special bucket called `example_partner_bucket` for Natalie to use.

Joe updates existing policies or adds new ones to accommodate the partner Widget Company. For example, Joe can create a new policy that denies members of the `WidgetCo` group the ability to use any actions other than write. This policy would be necessary only if there's a broad policy that gives all users access to a wide set of Amazon S3 actions.

The IAM Console and Sign-in Page

The AWS Management Console provides a web-based way to administer AWS services. You can sign in to the console and create, list, and perform other tasks with AWS services for your account. These tasks might include starting and stopping Amazon EC2 instances and Amazon RDS databases, creating Amazon DynamoDB tables, creating IAM users, and so on.

When you first create an Amazon Web Services (AWS) account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account.

Important

We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the [best practice of using the root user only to create your first IAM user \(p. 44\)](#). Then securely lock away the root user credentials and use them to perform only a few account and service management tasks. To view the tasks that require you to sign in as the root user, see [AWS Tasks That Require Root User](#). For a tutorial on how to set up an administrator for daily use, see [Creating Your First IAM Admin User and Group \(p. 17\)](#).

This section provides information about the AWS Management Console sign-in page. It explains how to create a unique sign-in URL for IAM users in your account, and how to sign in as the root user.

Note

If your organization has an existing identity system, you might want to create a single sign-on (SSO) option. SSO gives users access to the AWS Management Console for your account without requiring them to have an IAM user identity. SSO also eliminates the need for users to sign in to your organization's site and to AWS separately. For more information, see [Creating a URL that Enables Federated Users to Access the AWS Management Console \(Custom Federation Broker\) \(p. 169\)](#).

The IAM User Sign-in Page

To use the AWS Management Console, IAM users must provide their account ID or account alias in addition to their user name and password. When you, as an administrator, [create an IAM user in the console \(p. 64\)](#), you must send the sign-in credentials to that user, including the user name and the URL to the account sign-in page.

Important

Depending on how you set up the IAM user, provide all users with a temporary password for their first sign-in and, if appropriate, an MFA device. For detailed information about passwords and MFA devices, see [Managing Passwords \(p. 74\)](#) and [Using Multi-Factor Authentication \(MFA\) in AWS \(p. 91\)](#).

Your unique account sign-in page URL is created automatically when you begin using IAM. You do not have to do anything to use this sign-in page.

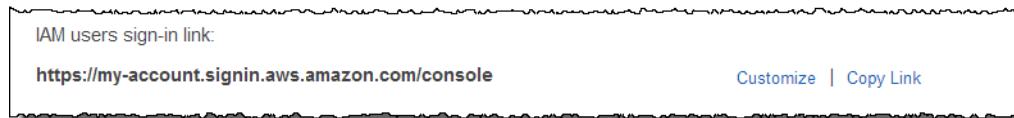
```
https://My_AWS_Account_ID.signin.aws.amazon.com/console/
```

You can also customize the account sign-in URL for your account if you want the URL to contain your company name (or other friendly identifier) instead of your AWS account ID number. For more information about creating an account alias, see [Your AWS Account ID and Its Alias \(p. 54\)](#).

Tip

To create a bookmark for your account sign-in page in your web browser, you should manually type the sign-in URL for your account in the bookmark entry. Do not use your web browser bookmark feature because redirects can obscure the sign-in URL.

You can find the URL for your account sign-in page anytime by viewing the dashboard of the IAM console.

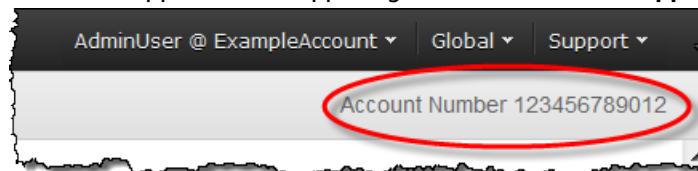


IAM users can also sign in at the following general sign-in endpoint and type an account ID or account alias manually:

<https://console.aws.amazon.com/>

Note

To find your AWS account ID number on the AWS Management Console, choose **Support** on the navigation bar on the upper-right, and then choose **Support Center**. Your currently signed-in account ID appears in the upper-right corner below the **Support** menu.



For convenience, the AWS sign-in page uses a browser cookie to remember the IAM user name and account information. The next time the user goes to any page in the AWS Management Console, the console uses the cookie to redirect the user to the account sign-in page.

The AWS Account Root User Sign-in Page

Use your AWS account email address and password to sign in to the [AWS Management Console](#) as the root user.

Note

If you previously signed in to the console with [IAM user \(p. 61\)](#) credentials, your browser might remember this preference and open your account-specific sign-in page. You cannot use the IAM user sign-in page to sign in with your AWS account root user credentials. If you see the IAM user sign-in page, choose **Sign-in using root account credentials** near the bottom of the page to return to the main sign-in page. From there, you can type your AWS account email address and password.

Controlling User Access to the AWS Management Console

Users who sign in to your AWS account through the sign-in page can access your AWS resources through the AWS Management Console to the extent that you grant them permission. The following list shows

the ways you can grant users access to your AWS account resources through the AWS Management Console. It also shows how users can access other AWS account features through the AWS website.

Note

There is no charge to use IAM.

The AWS Management Console

You create a password for each user who needs access to the AWS Management Console. Users access the console via your IAM-enabled AWS account sign-in page. For information about accessing the sign-in page, see [The IAM Console and Sign-in Page \(p. 52\)](#). For information about creating passwords, see [Managing Passwords \(p. 74\)](#).

Your AWS resources, such as Amazon EC2 instances, Amazon S3 buckets, and so on

Even if your users have passwords, they still need permission to access your AWS resources. When you create a user, that user has no permissions by default. To give your users the permissions they need, you attach policies to them. If you have many users who will be performing the same tasks with the same resources, you can assign those users to a group, then assign the permissions to that group. For information about creating users and groups, see [Identities \(Users, Groups, and Roles\) \(p. 59\)](#). For information about using policies to set permissions, see [Access Management \(p. 274\)](#).

AWS Discussion Forums

Anyone can read the posts on the [AWS Discussion Forums](#). Users who want to post questions or comments to the AWS Discussion Forum can do so using their user name. The first time a user posts to the AWS Discussion Forum, the user is prompted to enter a nickname and email address for use only by that user in the AWS Discussion Forums.

Your AWS account billing and usage information

You can grant users access your AWS account billing and usage information. For more information, see [Controlling Access to Your Billing Information](#) in the *AWS Billing and Cost Management User Guide*.

Your AWS account profile information

Users cannot access your AWS account profile information.

Your AWS account security credentials

Users cannot access your AWS account security credentials.

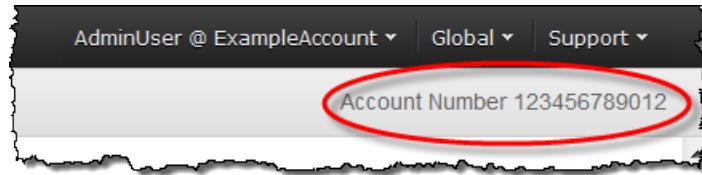
Note

IAM policies control access regardless of the interface. For example, you could provide a user with a password to access the AWS Management Console, and the policies for that user (or any groups the user belongs to) would control what the user can do in the AWS Management Console. Or, you could provide the user with AWS access keys for making API calls to AWS, and the policies would control which actions the user could call through a library or client that uses those access keys for authentication.

Your AWS Account ID and Its Alias

Finding Your AWS Account ID

To find your AWS account ID number on the AWS Management Console, choose **Support** on the navigation bar on the upper-right, and then choose **Support Center**. Your currently signed-in account ID appears in the upper-right corner below the **Support** menu.



About Account Aliases

If you want the URL for your sign-in page to contain your company name (or other friendly identifier) instead of your AWS account ID, you can create an alias for your AWS account ID. This section provides information about AWS account aliases and lists the API actions you use to create an alias.

Your sign-in page URL has the following format, by default.

```
https://Your_AWS_Account_ID.signin.aws.amazon.com/console/
```

If you create an AWS account alias for your AWS account ID, your sign-in page URL will look like the following example.

```
https://Your_Alias.signin.aws.amazon.com/console/
```

Note

The original URL containing your AWS account ID remains active and can be used after you create your AWS account alias.

Tip

To create a bookmark for your account sign-in page in your web browser, you should manually type the sign-in URL in the bookmark entry. Don't use your web browser's "bookmark this page" feature.

Creating, Deleting, and Listing an AWS Account Alias

You can use the AWS Management Console, the IAM API, or the command line interface to create or delete your AWS account alias.

Important

Your AWS account can have only one alias. If you create a new alias for your AWS account, the new alias overwrites the old alias, and the URL containing the old alias stops working.

AWS Management Console

To create or remove an account alias

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. On the navigation pane, select **Dashboard**.
3. Find the **IAM users sign-in link**, and click **Customize** to the right of the link.
4. Type the name you want to use for your alias, then click **Yes, Create**.
5. To remove the alias, click **Customize**, and then click **Yes, Delete**. The sign-in URL reverts to using your AWS account ID.

API or CLI

To create an alias for your AWS Management Console sign-in page URL

- AWS CLI: `aws iam create-account-alias`
- Tools for Windows PowerShell: [New-IAMAccountAlias](#)
- AWS API: [CreateAccountAlias](#)

To delete an AWS account ID alias

- AWS CLI: `aws iam delete-account-alias`
- Tools for Windows PowerShell: [Remove-IAMAccountAlias](#)
- AWS API: [DeleteAccountAlias](#)

To display your AWS account ID alias

- AWS CLI: `aws iam list-account-aliases`
- Tools for Windows PowerShell: [Get-IAMAccountAlias.html](#)
- AWS API: [ListAccountAliases](#)

Note

The account alias must be unique across all Amazon Web Services products. It must contain only digits, lowercase letters, and hyphens. For more information on limitations on AWS account entities, see [Limitations on IAM Entities and Objects \(p. 414\)](#).

Using MFA Devices With Your IAM Sign-in Page

IAM users who are configured with multi-factor authentication (MFA) devices must use their MFA devices to sign in to the AWS Management Console. After the user types the user name and password, AWS checks the user's account to see if MFA is required for that user. If so, a second sign-in page appears with an **MFA code** box to enter the numeric code provided by an MFA token device or sent to the user's mobile device as an SMS text message, depending on the type of MFA configured for the user.

If the MFA code is correct, then the user can access the AWS Management Console. If the code is incorrect, the user can try again with another code from a token device or by requesting that AWS send another SMS text message code. This is helpful if the first code was not received or does not work.

It's possible for an MFA token device to get out of synchronization. If after several unsuccessful tries a user cannot sign in to the AWS Management Console, the user is prompted to synchronize the MFA token device. The user can follow the on-screen prompts to synchronize the MFA token device. For information about how you can synchronize a device on behalf of a user in your AWS account, see [Resynchronize MFA Devices \(p. 102\)](#).

IAM Console Search

As you navigate through the IAM Management Console to manage various IAM resources, you often need to locate access keys or browse to the deeply nested IAM resources to find items you need to work with. A faster option is to use the IAM console search page to locate access keys related to your account, IAM entities (such as users, groups, roles, identity providers), policies by name, and more.

The IAM console search feature can locate any of the following:

- IAM entity names that match your search keywords (for users, groups, roles, identity providers, and policies)

- AWS documentation topic names that match your search keywords
- Tasks that match your search keywords

Every line in the search result is an active link. For example, you can choose the user name in the search result, which takes you to that user's detail page. Or you can choose an action link, for example **Create user**, to go to the **Create User** page.

Note

Access key search requires you to type the full access key ID in the search box. The search result shows the user associated with that key. From there you can navigate directly to that user's page, where you can manage their access key.

Using IAM Console Search

Use the **Search** page in the IAM console to find items related to that account.

To search for items in the IAM console

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, choose **Search**.
3. In the **Search** box, type your search keyword(s).
4. Choose a link in the search results list to navigate to the corresponding part of the console or documentation.

Icons in the IAM Console Search Results

The following icons identify the types of items that are found by a search:

Icon	Description
	IAM users
	IAM groups
	IAM roles
	IAM policies
	Tasks such as "create user" or "attach policy"
	Results from the keyword delete
	IAM documentation

Sample Search Phrases

You can use the following phrases in the IAM search. Replace terms in italics with the names of actual IAM users, groups, roles, access keys, policies, or identity providers respectively that you want to locate.

- `user_name` or `group_name` or `role_name` or `policy_name` or `identity_provider_name`
- `access_key`
- `add user user_name to groups` or `add users to group group_name`
- `remove user user_name from groups`
- `delete user_name` or `delete group_name` or `delete role_name`, or `delete policy_name`, or `delete identity_provider_name`
- `manage access keys user_name`
- `manage signing certificates user_name`
- `users`
- `manage MFA for user_name`
- `manage password for user_name`
- `create role`
- `password policy`
- `edit trust policy for role role_name`
- `show policy document for role role_name`
- `attach policy to role_name`
- `create managed policy`
- `create user`
- `create group`
- `attach policy to group_name`
- `attach entities to policy_name`
- `detach entities to policy_name`
- `what is IAM`
- `how do I create an IAM user`
- `how do I use IAM console`
- `what is a user` or `what is a group`, or `what is a policy`, or `what is a role`, or `what is an identity provider`

Identities (Users, Groups, and Roles)

This section describes *IAM identities*, which you create to provide authentication for people and processes in your AWS account. This section also describes *IAM groups*, which are collections of IAM users that you can manage as a unit. Identities represent the user, and can be authenticated and then authorized to perform actions in AWS. Each of these can be associated with one or more [policies \(p. 274\)](#) to determine what actions a user, role, or member of a group can do with which AWS resources and under what conditions.

The AWS Account Root User (p. 260)

When you first create an AWS account, you create an account (or root user) identity, which you use to sign in to AWS. You can sign in to the AWS Management Console as the root user—that is, the email address and password that you provide when you create the account. This combination of your email address and password is called your root user credentials.

When you sign in as the root user, you have complete, unrestricted access to all resources in your AWS account, including access to your billing information and the ability to change your password. This level of access is necessary when you initially set up the account. However, we recommend that you **don't** use root user credentials for everyday access. We especially recommend that you do not share your root user credentials with anyone, because doing so gives them unrestricted access to your account. It is not possible to restrict the permissions that are granted to the AWS account. Instead, we strongly recommend that you adhere to the [best practice of using the root user only to create your first IAM user \(p. 44\)](#) and then securely locking away the root user credentials.

IAM Users (p. 61)

An IAM [user \(p. 61\)](#) is an entity that you create in AWS. The IAM user represents the person or service who uses the IAM user to interact with AWS. A primary use for IAM users is to give people the ability to sign in to the AWS Management Console for interactive tasks and to make programmatic requests to AWS services using the API or CLI. A user in AWS consists of a name, a password to sign into the AWS Management Console, and up to two access keys that can be used with the API or CLI. When you create an IAM user, you grant it permissions by making it a member of a group that has appropriate permission policies attached (recommended), or by directly attaching policies to the user. You can also clone the permissions of an existing IAM user, which automatically makes the new user a member of the same groups and attaches all the same policies.

IAM Groups (p. 129)

An IAM [group \(p. 129\)](#) is a collection of IAM users. You can use groups to specify permissions for a collection of users, which can make those permissions easier to manage for those users. For example, you could have a group called *Admins* and give that group the types of permissions that administrators typically need. Any user in that group automatically has the permissions that are assigned to the group. If a new user joins your organization and should have administrator privileges, you can assign the appropriate permissions by adding the user to that group. Similarly, if a person changes jobs in your organization, instead of editing that user's permissions, you can remove him or her from the old groups and add him or her to the appropriate new groups. Note that a group is not truly an identity because it

cannot be identified as a **Principal** in a permission policy. It is only a way to attach policies to multiple users at one time.

IAM Roles (p. 135)

An IAM [role \(p. 135\)](#) is very similar to a user, in that it is an identity with permission policies that determine what the identity can and cannot do in AWS. However, a role does not have any credentials (password or access keys) associated with it. Instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. An IAM user can assume a role to temporarily take on different permissions for a specific task. A role can be assigned to a [federated user \(p. 143\)](#) who signs in by using an external identity provider instead of IAM. AWS uses details passed by the identity provider to determine which role is mapped to the federated user.

Temporary Credentials (p. 231)

Temporary credentials are primarily used with IAM roles, but there are also other uses. You can request temporary credentials that have a more restricted set of permissions than your standard IAM user. This prevents you from accidentally performing tasks that are not permitted by the more restricted credentials. A benefit of temporary credentials is that they expire automatically after a set period of time. You have control over the duration that the credentials are valid.

When to Create an IAM User (Instead of a Role)

Because an IAM user is just an identity with specific permissions in your account, you might not need to create an IAM user for every occasion on which you need credentials. In many cases, you can take advantage of IAM *roles* and their temporary security credentials instead of using the long-term credentials associated with an IAM user.

- **You created an AWS account and you're the only person who works in your account.**

It's possible to work with AWS using the root user credentials for your AWS account, but we don't recommend it. Instead, we strongly recommend that you create an IAM user for yourself and use the credentials for that user when you work with AWS. For more information, see [IAM Best Practices \(p. 43\)](#).

- **Other people in your group need to work in your AWS account, and your group is using no other identity mechanism.**

Create IAM users for the individuals who need access to your AWS resources, assign appropriate permissions to each user, and give each user his or her own credentials. We strongly recommend that you never share credentials among multiple users.

- **You want to use the command-line interface (CLI) to work with AWS.**

The CLI needs credentials that it can use to make calls to AWS. Create an IAM user and give that user permissions to run the CLI commands you need. Then configure the CLI on your computer to use the access key credentials associated with that IAM user.

When to Create an IAM Role (Instead of a User)

Create an IAM role in the following situations:

You're creating an application that runs on an Amazon Elastic Compute Cloud (Amazon EC2) instance and that application makes requests to AWS.

Don't create an IAM user and pass the user's credentials to the application or embed the credentials in the application. Instead, create an IAM role that you attach to the EC2 instance to give applications running on the instance temporary security credentials. The credentials have the permissions specified in the policies attached to the role. For details, see [Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances \(p. 215\)](#).

You're creating an app that runs on a mobile phone and that makes requests to AWS.

Don't create an IAM user and distribute the user's access key with the app. Instead, use an identity provider like Login with Amazon, Amazon Cognito, Facebook, or Google to authenticate users and map the users to an IAM role. The app can use the role to get temporary security credentials that have the permissions specified by the policies attached to the role. For more information, see the following:

- [Amazon Cognito Overview in the AWS Mobile SDK for Android Developer Guide](#)
- [Amazon Cognito Overview in the AWS Mobile SDK for iOS Developer Guide](#)
- [About Web Identity Federation \(p. 143\)](#)

Users in your company are authenticated in your corporate network and want to be able to use AWS without having to sign in again—that is, you want to allow users to federate into AWS.

Don't create IAM users. Configure a federation relationship between your enterprise identity system and AWS. You can do this in two ways:

- If your company's identity system is compatible with SAML 2.0, you can establish trust between your company's identity system and AWS. For more information, see [About SAML 2.0-based Federation \(p. 149\)](#).
- Create and use a custom proxy server that translates user identities from the enterprise into IAM roles that provide temporary AWS security credentials. For more information, see [Creating a URL that Enables Federated Users to Access the AWS Management Console \(Custom Federation Broker\) \(p. 169\)](#).

IAM Users

An IAM *user* is an entity that you create in AWS to represent the person or service that uses it to interact with AWS. A user in AWS consists of a name and credentials.

An IAM user with administrator permissions is not the same thing as the AWS account root user. For more information about the root user, see [The AWS Account Root User \(p. 260\)](#).

Important

If you arrived at this page while trying to enable Amazon Advertising for your application or web site, see [Becoming a Product Advertising API Developer](#).

How AWS identifies an IAM user

When you create a user, IAM creates these ways to identify that user:

- A "friendly name" for the user, which is the name that you specified when you created the user, such as Bob or Alice. These are the names you see in the AWS Management Console.
- An Amazon Resource Name (ARN) for the user. You use the ARN when you need to uniquely identify the user across all of AWS, such as when you specify the user as a Principal in an IAM policy for an Amazon S3 bucket. An ARN for an IAM user might look like the following:

`arn:aws:iam::account-ID-without-hyphens:user/Bob`

- A unique identifier for the user. This ID is returned only when you use the API, Tools for Windows PowerShell, or AWS CLI to create the user; you do not see this ID in the console.

For more information about these identifiers, see [IAM Identifiers \(p. 410\)](#).

Users and credentials

You can access AWS in different ways depending on the user credentials:

- [Console password \(p. 74\)](#): A password that the user can type to sign in to interactive sessions such as the AWS Management Console.
- [Access keys \(p. 85\)](#): A combination of an access key ID and a secret access key. You can assign two to a user at a time. These can be used to make programmatic calls to AWS when using the API in program code or at a command prompt when using the AWS CLI or the AWS PowerShell tools.
- [SSH keys for use with AWS CodeCommit \(p. 123\)](#): An SSH public key in the OpenSSH format that can be used to authenticate with AWS CodeCommit.
- [Server certificates \(p. 125\)](#): SSL/TLS certificates that you can use to authenticate with some AWS services. We recommend that you use AWS Certificate Manager (ACM) to provision, manage, and deploy your server certificates. Use IAM only when you must support HTTPS connections in a region that is not supported by ACM. To learn which regions support ACM, see the [AWS Certificate Manager](#) section of the [AWS General Reference](#).

By default, a brand new IAM user has no password and no access key (neither an access key ID nor a secret access key)—no credentials of any kind. You must create the type of credentials for an IAM user based on what the user will be doing.

Take advantage of the following options to administer passwords, access keys, and MFA devices:

- [Manage passwords for your IAM users \(p. 74\)](#). Create and change the passwords that permit access to the AWS Management Console. Set a password policy to enforce a minimum password complexity. Allow users to change their own passwords.
- [Manage access keys for your IAM users \(p. 85\)](#). Create and update access keys for programmatic access to the resources in your account.
- You can enhance the security of the user's credentials by enabling [multi-factor authentication \(MFA\) \(p. 91\)](#) for the user. With MFA, users have to provide both the credentials that are part of their user identity (a password or access key) and a temporary numeric code that's generated on a hardware device or by an application on a smartphone or tablet, or sent by AWS to an SMS-compatible mobile device.
- [Find unused passwords and access keys \(p. 117\)](#). Anyone who has a password or access keys for your account or an IAM user in your account has access to your AWS resources. The security [best practice](#) is to remove passwords and access keys when users no longer need them.
- [Download a credential report for your account \(p. 120\)](#). You can generate and download a credential report that lists all IAM users in your account and the status of their various credentials, including passwords, access keys, and MFA devices. For passwords and access keys, the credential report shows how recently the password or access key has been used.

Users and permissions

By default, a brand new IAM user has no [permissions \(p. 274\)](#) to do anything. The user is not authorized to perform any AWS actions or to access any AWS resources. An advantage of having individual IAM users

is that you can assign permissions individually to each user. You might assign administrative permissions to a few users, who then can administer your AWS resources and can even create and manage other IAM users. In most cases, however, you want to limit a user's permissions to just the tasks (AWS actions) and resources that the user needs for his or her job. Imagine a user named Dave. When you create the IAM user Dave, you create a password for that user and attach permissions to the IAM user that let him launch a specific Amazon EC2 instance and read (GET) information from a table in an Amazon RDS database. For procedures on how to create users and grant them initial credentials and permissions, see [Creating an IAM User in Your AWS Account \(p. 63\)](#). For procedures on how to change the permissions for existing users, see [Changing Permissions for an IAM User \(p. 71\)](#). For procedures on how to change the user's password or access keys, see [Managing Passwords \(p. 74\)](#) and [Managing Access Keys for IAM Users \(p. 85\)](#).

Users and accounts

Each IAM user is associated with one and only one AWS account. Because users are defined within your AWS account, they don't need to have a payment method on file with AWS. Any AWS activity performed by users in your account is billed to your account.

There's a limit to the number of IAM users you can have in an AWS account. For more information, see [Limitations on IAM Entities and Objects \(p. 414\)](#).

Users as service accounts

An IAM user doesn't necessarily have to represent an actual person. An IAM user is really just an identity with associated credentials and permissions. You might create an IAM user to represent an application that needs credentials in order to make requests to AWS. This is typically referred to as a "service account." An application might have its own service account in your AWS account, and its own set of permissions, the same way that processes have their own service accounts defined in an operating system like Windows or Linux.

Creating an IAM User in Your AWS Account

 Follow us on Twitter

You can create one or more IAM users in your AWS account. You might create an IAM user when someone joins your organization, or when you have a new application that needs to make API calls to AWS.

Important

If you arrived at this page trying to enable Amazon Advertising for your application or website, see [Becoming a Product Advertising API Developer](#).

If you arrived at this page from the IAM console, it is possible that your account does not include IAM users, even though you are logged in. You could be signed in as the AWS account root user, using a role, or signed in with temporary credentials. To learn more about these IAM identities, see [Identities \(Users, Groups, and Roles\) \(p. 59\)](#).

Topics

- [Creating IAM Users \(Console\) \(p. 64\)](#)
- [Creating IAM Users \(AWS CLI, Tools for Windows PowerShell, or IAM HTTP API\) \(p. 66\)](#)

In outline, the process of creating a user and making it usable for work tasks consists of these steps:

1. Create the user in the AWS Management Console or from an AWS CLI, Tools for Windows PowerShell, or IAM API command. If you create the user in the AWS Management Console, then steps 1–4 are handled automatically. If you create the users programmatically, then you must perform each of those steps individually.
2. Create credentials for the user, depending on the type of access the user requires:

- **Programmatic access:** The IAM user might need to make API calls or use the AWS CLI or the Tools for Windows PowerShell. In that case, create an access key (an access key ID and a secret access key) for that user.

AWS Management Console access: If the user needs to access AWS resources from the AWS Management Console, [create a password for the user \(p. 79\)](#).

As a best practice, do not create credentials of a certain type for a user who will never need that kind of access. For example, for a user who requires access through the AWS Management Console only, do not create access keys.

3. Give the user permissions to perform the required tasks by adding the user to one or more groups. You can grant permissions by attaching IAM permission policies directly to the user. However, we recommend instead that you put your users in groups and manage permissions through policies that are attached to those groups.
4. Provide the user with the necessary sign-in information. This includes the password and the URL for the account sign-in webpage where the user enters those credentials. For more information, see [How IAM Users Sign In to AWS \(p. 67\)](#).
5. (Optional) Configure [multi-factor authentication \(MFA\) \(p. 91\)](#) for the user. MFA requires the user to provide a one-time-use code each time he or she signs into the AWS Management Console.
6. (Optional) Give users permissions to manage their own security credentials. (By default, users do not have permissions to manage their own credentials.) For more information, see [Permitting IAM Users to Change Their Own Passwords \(p. 82\)](#).

For information about the permissions that you need in order to create a user, see [Permissions Required to Access IAM Resources \(p. 372\)](#).

Creating IAM Users (Console)

To create one or more IAM users from the AWS Management Console

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, choose **Users** and then choose **Add user**.
3. Type the user name for the new user. This is the sign-in name for AWS. If you want to add more than one user at the same time, choose **Add another user** for each additional user and type their user names. You can add up to 10 users at one time.

Note

User names can be a combination of up to 64 letters, digits, and these characters: plus (+), equal (=), comma (,), period (.), at sign (@), and hyphen (-). Names must be unique within an account. They are not distinguished by case. For example, you cannot create two users named *TESTUSER* and *testuser*. For more information about limitations on IAM entities, see [Limitations on IAM Entities and Objects \(p. 414\)](#).

4. Select the type of access this set of users will have. You can select programmatic access, access to the AWS Management Console, or both.
 - Select **Programmatic access** if the users require access to the API, AWS CLI, or Tools for Windows PowerShell. This creates an access key for each new user. You can view or download the access keys when you get to the **Final** page.
 - Select **AWS Management Console access** if the users require access to the AWS Management Console. This creates a password for each new user.

1. For **Console password type**, choose one of the following:
 - **Autogenerated password.** Each user gets a randomly generated password that meets the current password policy in effect (if any). You can view or download the passwords when you get to the **Final** page.
 - **Custom password.** Each user is assigned the password that you type in the box.
2. (Optional) We recommend that you choose **Require password reset** to ensure that users are forced to change their password the first time they sign in.

Note

If you have *not* enabled [the account-wide password policy setting Allow users to change their own password](#), then selecting **Require password reset** automatically attaches an AWS managed policy named [IAMUserChangePassword](#) to the new users that grants them permission to change their own passwords.

5. Choose **Next: Permissions**.
6. On the **Set permissions** page, specify how you want to assign permissions to this set of new users. Choose one of the following three options:
 - **Add user to group.** Choose this option if you have groups with appropriate permission policies already created and want to assign the users to those groups. IAM displays a list of all currently defined groups, along with their attached policies. You can select one or more existing groups, or choose **Create group** to create a new group. For more information, see [Changing Permissions for an IAM User \(p. 71\)](#).
 - **Copy permissions from existing user.** Choose this option to copy all of the group memberships, attached managed policies, and embedded inline policies from an existing user to the new users. IAM displays a list of currently defined users. Select the one whose permissions most closely match the needs of your new users. Each new user gets the same group memberships and attached policies as the selected user.
 - **Attach existing policies to user directly** Choose this option to select from existing managed policies or to create new managed policies that are attached to the new users. IAM displays a list of currently defined managed policies, both AWS and customer-defined. Select the policies that you want to attach to the new users or choose **Create policy** to create a new policy from scratch. For more information, see step 4 in the procedure [Create an IAM Policy \(Console\) \(p. 317\)](#).
7. Choose **Next: Review** to see all of the choices you made up to this point. When you are ready to proceed, choose **Create user**.
8. To view the users' access keys (access key IDs and secret access keys), choose **Show** next to each password and secret access key that you want to see. To save the access keys, choose **Download .csv** and then save the file to a safe location.

Important

This is your only opportunity to view or download the secret access keys, and you must provide this information to your users before they can use the AWS API. Save the user's new access key ID and secret access key in a safe and secure place. **You will not have access to the secret keys again after this step.**

9. Provide each user with his or her credentials. On the final page you can choose **Send email** next to each user. Your local mail client opens with a draft that you can customize and send. The email template includes the following details to each user:
 - User name

- URL to the account sign-in webpage. Use the following example, substituting the correct account ID number or account alias:

```
https://AWS-account-ID or alias.signin.aws.amazon.com/console
```

For more information, see [How IAM Users Sign In to AWS \(p. 67\)](#).

Important

The user's password is **not** included in the generated email. You must provide them to the customer in a way that complies with your organization's security guidelines.

10. (Optional) Grant the users permission to manage their own security credentials. For more information, see [Allow Users to Manage Their Own Passwords, Access Keys, and SSH Keys \(p. 377\)](#).

Creating IAM Users (AWS CLI, Tools for Windows PowerShell, or IAM HTTP API)

To create an IAM user from the AWS CLI, Tools for Windows PowerShell, or IAM HTTP API

1. **Create a user.**

- AWS CLI: [aws iam create-user](#)
- Tools for Windows PowerShell: [New-IAMUser](#)
- IAM API: [CreateUser](#)

2. (Optional) **Give the user access to the AWS Management Console.** This requires a password. You must also give the user the [URL of your account's sign-in page. \(p. 67\)](#)

- AWS CLI: [aws iam create-login-profile](#)
- Tools for Windows PowerShell: [New-IAMLoginProfile](#)
- IAM API: [CreateLoginProfile](#)

3. (Optional) **Give the user programmatic access.** This requires access keys.

- AWS CLI: [aws iam create-access-key](#)
- Tools for Windows PowerShell: [New-IAMAccessKey](#)
- IAM API: [CreateAccessKey](#)

Important

This is your only opportunity to view or download the secret access keys, and you must provide this information to your users before they can use the AWS API. Save the user's new access key ID and secret access key in a safe and secure place. **You will not have access to the secret keys again after this step.**

4. **Add the user to one or more groups.** The groups that you specify should have attached policies that grant the appropriate permissions for the user.

- AWS CLI: [aws iam add-user-to-group](#)
- Tools for Windows PowerShell: [Add-IAMUserToGroup](#)
- IAM API: [AddUserToGroup](#)

5. (Optional) **Attach a policy to the user** Attach a policy that defines the user's permissions. **Note:** We recommend that you manage user permissions by adding the user to a group and attaching a policy to the group instead of attaching directly to a user.

- AWS CLI: [aws iam attach-user-policy](#)
- Tools for Windows PowerShell: [Register-IAMUserPolicy](#)

- IAM API: [AttachUserPolicy](#)
6. (Optional) **Give the user permission to manage his or her own security credentials.** For more information, see [Allow Users to Manage Their Own Passwords, Access Keys, and SSH Keys \(p. 377\)](#).

How IAM Users Sign In to AWS

To sign in to the AWS Management Console as an IAM user, you must provide your account ID or account alias in addition to your user name and password. When your administrator [created your IAM user in the console \(p. 64\)](#), they should have sent you your sign-in credentials, including your user name and the URL to your account sign-in page that includes your account ID or account alias.

```
https://My_AWS_Account_ID.signin.aws.amazon.com/console/
```

Tip

To create a bookmark for your account sign-in page in your web browser, you should manually type the sign-in URL for your account in the bookmark entry. Do not use your web browser bookmark feature because redirects can obscure the sign-in URL.

You can also sign in at the following general sign-in endpoint and type your account ID or account alias manually:

```
https://console.aws.amazon.com/
```

For convenience, the AWS sign-in page uses a browser cookie to remember the IAM user name and account information. The next time the user goes to any page in the AWS Management Console, the console uses the cookie to redirect the user to the account sign-in page.

You have access only to the AWS resources that your administrator specifies in the policy that is attached to your IAM user identity. To work in the console, you must have permissions to perform the actions that the console performs, such as listing and creating AWS resources. For more information, see [Access Management \(p. 274\)](#) and [Example Policies \(p. 295\)](#).

Note

If your organization has an existing identity system, you might want to create a single sign-on (SSO) option. SSO gives users access to the AWS Management Console for your account without requiring them to have an IAM user identity. SSO also eliminates the need for users to sign in to your organization's site and to AWS separately. For more information, see [Creating a URL that Enables Federated Users to Access the AWS Management Console \(Custom Federation Broker\) \(p. 169\)](#).

Logging sign-in details in CloudTrail

If you enable CloudTrail to log sign-in events to your logs, you need to be aware of how CloudTrail chooses where to log the events.

- If your users sign-in directly to a console, they are redirected to either a global or a regional sign-in endpoint, based on whether the selected service console supports regions. For example, the main console home page supports regions, so if you sign in to the following URL:

```
https://alias.signin.aws.amazon.com/console
```

you are redirected to a regional sign-in endpoint such as `https://us-east-2.signin.aws.amazon.com`, resulting in a regional CloudTrail log entry in the user's region's log:

On the other hand, the Amazon S3 console does not support regions, so if you sign in to the following URL

```
https://alias.signin.aws.amazon.com/console/s3
```

AWS redirects you to the global sign-in endpoint at `https://signin.aws.amazon.com`, resulting in a global CloudTrail log entry.

- You can manually request a certain regional sign-in endpoint by signing in to the region-enabled main console home page using a URL syntax like the following:

```
https://alias.signin.aws.amazon.com/console?region=ap-southeast-1
```

AWS redirects you to the `ap-southeast-1` regional sign-in endpoint and results in a regional CloudTrail log event.

For more information about CloudTrail and IAM, see [Logging IAM Events with AWS CloudTrail](#).

If users need programmatic access to work with your account, you can create an access key pair (an access key ID and a secret access key) for each user, as described in [Creating, Modifying, and Viewing Access Keys \(Console\) \(p. 85\)](#).

Managing IAM Users

Amazon Web Services offers multiple tools for managing the IAM users in your AWS account.

Topics

- [Listing IAM Users \(p. 68\)](#)
- [Renaming an IAM User \(p. 69\)](#)
- [Deleting an IAM User \(p. 69\)](#)

Listing IAM Users

You can list the IAM users in your AWS account or in a specific IAM group, and list all the groups that a user is in. For information about the permissions that you need in order to list users, see [Permissions Required to Access IAM Resources \(p. 372\)](#).

To list all the users in the account

- [AWS Management Console](#): In the navigation pane, choose **Users**. The console displays the users in your AWS account.
- [AWS CLI](#): `aws iam list-users`
- Tools for Windows PowerShell: [Get-IAMUsers](#)
- AWS API: [ListUsers](#)

To list the users in a specific group

- [AWS Management Console](#): In the navigation pane, choose **Groups**, choose the name of the group, and then choose the **Users** tab.
- [AWS CLI](#): `aws iam get-group`
- Tools for Windows PowerShell: [Get-IAMGroup](#)

- AWS API: [GetGroup](#)

To list all the groups that a user is in

- **AWS Management Console:** In the navigation pane, choose **Users**, choose the user name, and then choose the **Groups** tab.
- AWS CLI: `aws iam list-groups-for-user`
- Tools for Windows PowerShell: `Get-IAMGroupForUser`
- AWS API: [ListGroupsForUser](#)

Renaming an IAM User

To change a user's name or path, you must use the AWS CLI, Tools for Windows PowerShell, or AWS API. There is no option in the console to rename a user. For information about the permissions that you need in order to rename a user, see [Permissions Required to Access IAM Resources \(p. 372\)](#).

When you change a user's name or path, the following happens:

- Any policies attached to the user stay with the user under the new name.
- The user stays in the same groups under the new name.
- The unique ID for the user remains the same. For more information about unique IDs, see [Unique IDs \(p. 413\)](#).
- Any resource or role policies that refer to the user *as a principal* (the user is being granted access) are automatically updated to use the new name or path. For example, any queue-based policies in Amazon SQS or resource-based policies in Amazon S3 are automatically updated to use the new name and path.

IAM does not automatically update policies that refer to the user *as a resource* to use the new name or path; you must manually do that. For example, imagine that user Bob has a policy attached to him that lets him manage his security credentials. If an administrator renames Bob to Robert, the administrator also needs to update that policy to change the resource from this:

```
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/Bob
```

to this:

```
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/Robert
```

This is true also if an administrator changes the path; the administrator needs to update the policy to reflect the new path for the user.

To rename a user

- AWS CLI: `aws iam update-user`
- Tools for Windows PowerShell: `Update-IAMUser`
- AWS API: [UpdateUser](#)

Deleting an IAM User

You might delete an IAM user from your account if someone quits your company. If the user is only temporarily away from your company, you can disable the user's credentials instead of deleting the user.

entirely from the AWS account. That way, you can prevent the user from accessing the AWS account's resources during the absence but you can re-enable the user later.

For more information about disabling credentials, see [Managing Access Keys for IAM Users \(p. 85\)](#). For information about the permissions that you need in order to delete a user, see [Permissions Required to Access IAM Resources \(p. 372\)](#).

Topics

- [Deleting an IAM User \(AWS Management Console\) \(p. 70\)](#)
- [Deleting an IAM User \(AWS CLI and Tools for Windows PowerShell\) \(p. 70\)](#)

Deleting an IAM User (AWS Management Console)

When you use the AWS Management Console to delete an IAM user, IAM automatically deletes the following information for you:

- The user
- Any group memberships—that is, the user is removed from any IAM groups that the user was a member of
- Any password associated with the user
- Any access keys belonging to the user
- All inline policies embedded in the user (policies that are applied to a user via group permissions are not affected)

Note

Any managed policies attached to the user are detached from the user when the user is deleted. Managed policies are not deleted when you delete a user.

- Any associated MFA device

To use the AWS Management Console to delete an IAM user

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, choose **Users**, and then select the check box next to the user name that you want to delete, not the name or row itself.
3. At the top of the page, choose **Delete user**.
4. In the confirmation dialog box, wait for the service last accessed data to load before you review the data. The dialog box shows when each of the selected users last accessed an AWS service. If you attempt to delete a user that has been active within the last 30 days, you must select an additional check box to confirm that you want to delete the active user. If you want to proceed, choose **Yes, Delete**.

Deleting an IAM User (AWS CLI and Tools for Windows PowerShell)

Unlike the AWS Management Console, when you delete a user with the AWS CLI or Tools for Windows PowerShell you have to delete the items attached to the user manually. This procedure illustrates the process. For a complete PowerShell code snippet, see the example in [Remove-IAMUser](#).

To use the AWS CLI to delete a user from your account

1. Delete the user's keys and certificates. This helps ensure that the user can't access your AWS account's resources anymore. Note that when you delete a security credential, it's gone forever and can't be retrieved.

- aws iam delete-access-key and aws iam delete-signing-certificate
2. Delete the user's password, if the user has one.
[aws iam delete-login-profile](#)
3. Deactivate the user's MFA device, if the user has one.
[aws iam deactivate-mfa-device](#)
4. Detach any policies that are attached to the user.
[aws iam list-attached-user-policies](#) (to list the policies attached to the user) and [aws iam detach-user-policy](#) (to detach the policies)
5. Get a list of any groups the user was in, and remove the user from those groups.
[aws iam list-groups-for-user](#) and [aws iam remove-user-from-group](#)
6. Delete the user.
[aws iam delete-user](#)

To use the Tools for Windows PowerShell to delete a user from your account

1. Delete the user's keys and certificates. This helps ensure that the user can't access your AWS account's resources anymore. Note that when you delete a security credential, it's gone forever and can't be retrieved.
[Remove-IAMAccessKey](#) and [Remove-IAMSigningCertificate](#)
2. Delete the user's password, if the user has one.
[Remove-IAMLoginProfile](#)
3. Deactivate the user's MFA device, if the user has one.
[Disable-IAMMFADevice](#)
4. Detach any policies that are attached to the user.
[Get-IAMAttachedUserPolicies](#) (to list the policies attached to the user) and [Remove-IAMUserPolicy](#) (to detach the policies).
5. Get a list of any groups the user was in, and remove the user from those groups.
[Get-IAMGroupForUser](#) and [Remove-IAMUserFromGroup](#).
6. Delete the user.
[Remove-IAMUser](#)

Changing Permissions for an IAM User

You can change the permissions for an IAM user in your AWS account by changing its group memberships or by attaching and detaching managed policies. A user gets its permissions through one of the following methods:

Group membership

- Add or remove a user from a group.
- Add, remove, or edit a managed policy attached to the group. This policy can be an [AWS managed policy \(p. 280\)](#) or a [customer managed policy \(p. 281\)](#) that you create.

- Add, remove, or edit a group's [inline policies \(p. 282\)](#).

Direct policy attachment

- Add, remove, or edit a managed policy attached directly to a user. This policy can be an [AWS managed policy \(p. 280\)](#) or a [customer managed policy \(p. 281\)](#) that you create.
- Add, remove, or edit a user's [inline policies \(p. 282\)](#).

For information about the permissions that you need in order to modify the permissions for a user, see [Permissions Required to Access IAM Resources \(p. 372\)](#).

Adding Permissions to a New or Existing User (Console)

You can change permissions associated with a user through one of three techniques:

- [Add user to group \(p. 72\)](#) – Make the user a member of a group that already has policies attached. Every member of the group receives the permissions granted by the group's policies.
- [Copy permissions from existing user \(p. 73\)](#) – Copy all group memberships and attached managed policies as well as all inline policies embedded in the source user.
- [Attach policies directly to user \(p. 73\)](#) – Attach a managed policy directly to the user. As a [best practice \(p. 44\)](#), we recommend that you instead attach your policies to a group and then make users members of the appropriate groups.

Adding Permissions by Adding the User to a Group

To add permissions to a user by adding the user to a group

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, choose **Users**.
3. Review the current group memberships for users in the **Groups** column of the console. If necessary, add the column to the users table by completing the following steps:
 1. Above the table on the far right, choose the settings symbol ().
 2. In the **Manage Columns** dialog box, select the **Groups** column. Optionally, you can also clear the check box for any column headings that you do not want to appear in the users table.
 3. Choose **Close** to return to the list of users.

The **Groups** column tells you to which groups the user belongs. The field includes the group names for up to two groups. If the user is a member of three or more groups, the first two groups are shown (ordered alphabetically), and the number of additional group memberships is included. For example, if the user belongs to Group A, Group B, Group C, and Group D, then the field contains the value **Group A, Group B + 2 more**. To see the total number of groups to which the user belongs, you can add the **Group count** column to the users table.

4. Choose the name of the user whose permissions you want to modify.
5. Choose the **Permissions** tab, and then choose **Add permissions**. Under **Grant permissions** choose **Add user to group**.
6. Select the check box for each group that you want the user to join. The list shows each group's name and the policies that the user receives if made a member of that group. The permissions in each selected group apply to the user as soon as you complete the process.
7. (Optional) In addition to selecting from existing groups, you can choose **Create group** to define a new group:

- a. For **Group name**, type a name for your new group.

Note

Group names can be a combination of up to 128 letters, digits, and these characters: plus (+), equal (=), comma (,), period (.), at sign (@), and hyphen (-). Names must be unique within an account. They are not distinguished by case. For example, you cannot create two groups named *TESTGROUP* and *testgroup*. For more information about limitations on IAM entities, see [Limitations on IAM Entities and Objects \(p. 414\)](#).

- b. Select one or more check boxes for the managed policies that you want to attach to the group. You can also create a new managed policy by choosing **Create policy**. If you do, return to this browser tab or window when the new policy is done; choose **Refresh**; and then choose the new policy to attach it to your group. For more information, see [Creating IAM Policies \(p. 317\)](#).
 - c. Choose **Create group**.
 - d. Back in the list of groups, select the check box for your new group.
8. Choose **Next: Review** to see the list of group memberships to be added to the user. Then choose **Add permissions**.

The new permissions affect the user immediately.

Adding Permissions by Copying from Another User

To add permissions to a user by copying permissions from another user

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Choose **Users** in the navigation pane, choose the name of the user whose permissions you want to modify, and then choose the **Permissions** tab.
3. Choose **Add permissions**, and then under **Grant permissions** choose **Copy permissions from existing user**. The list displays available users along with their group memberships and attached policies. If the full list of groups or policies don't fit on one line, you can choose the link for **and n more**. Doing that opens a new browser tab and see the full list of policies (**Permissions** tab) and groups (**Groups** tab).
4. Select the radio button next to the user whose permissions you want to copy.
5. Choose **Next: Review** to see the list of changes that are to be made to the user. Then choose **Add permissions**.

The new permissions affect the user immediately.

Adding Permissions by Attaching Policies Directly to the User

To add permissions to a user by directly attaching managed policies

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Choose **Users** in the navigation pane, choose the name of the user whose permissions you want to modify, and then choose the **Permissions** tab.
3. Choose **Add permissions**, and then under **Grant permissions**, choose **Attach existing policies directly to user**.
4. Select one or more check boxes for the managed policies that you want to attach to the group. You can also create a new managed policy by choosing **Create policy**. If you do, return to this browser tab or window when the new policy is done. Choose **Refresh**; and then select the check box for the new policy to attach it to your user. For more information, see [Creating IAM Policies \(p. 317\)](#).

5. Choose **Next: Review** to see the list of policies that are to be attached to the user. Then choose **Add permissions**.

The new permissions affect the user immediately.

Removing Permissions from an Existing User (Console)

To revoke permissions for IAM users

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, choose **Users**, choose the name of the user whose permissions you want to modify, and then choose the **Permissions** tab.
- The **Permissions** tab displays each policy that applies to the user, and how the user gets that policy.
3. If you want to revoke permissions by removing an existing policy, view the **Policy type** to understand how the user is getting that policy before choosing **X** to remove the policy:
 - If the policy applies because of group membership, then choosing **X** removes the user from the group. Because you might have multiple policies attached from a single group, if you remove a user from a group, the user loses access to *all* policies that it received through that group membership..
 - If the policy is a managed policy attached directly to the user, then choosing **X** detaches the policy from the user. This does not affect the policy itself or any other entity that the policy might be attached to.
 - If the policy is an inline embedded policy, then choosing **X** removes the policy from IAM. Inline policies that are attached directly to a user exist only on that user.

Adding and Removing Permissions from a User (AWS API, AWS CLI, Tools for Windows PowerShell)

To add or remove permissions programmatically, you must add or remove the group memberships, attach or detach the managed policies, or add or delete the inline policies. For more information, see the following topics:

- [Adding and Removing Users in an IAM Group \(p. 132\)](#)
- [Attaching or Detaching IAM Policies \(AWS CLI or AWS API\) \(p. 333\)](#)
- [Attaching and Detaching IAM Policies \(p. 331\)](#)

Managing Passwords

You can manage passwords for your AWS account root user and for IAM users in your account. IAM users need passwords in order to access the AWS Management Console. Users do not need passwords to access AWS resources programmatically by using the AWS CLI, Tools for Windows PowerShell, the AWS SDKs or APIs. For those environments, users need [access keys \(p. 85\)](#) instead.

Topics

- [Changing the AWS Account Root User Password \(p. 75\)](#)
- [Setting an Account Password Policy for IAM Users \(p. 75\)](#)
- [Managing Passwords for IAM Users \(p. 79\)](#)
- [Permitting IAM Users to Change Their Own Passwords \(p. 82\)](#)

- [How IAM Users Change Their Own Password \(p. 83\)](#)

Changing the AWS Account Root User Password

You must be signed in as the AWS account root user in order to change the password. To learn how to reset a forgotten root user password, see [Resetting Your Lost or Forgotten Passwords or Access Keys \(p. 90\)](#).

To change the password for the root user

1. Use your AWS account email address and password to sign in to the [AWS Management Console](#) as the root user.

Note

If you previously signed in to the console with [IAM user \(p. 61\)](#) credentials, your browser might remember this preference and open your account-specific sign-in page. You cannot use the IAM user sign-in page to sign in with your AWS account root user credentials. If you see the IAM user sign-in page, choose **Sign-in using root account credentials** near the bottom of the page to return to the main sign-in page. From there, you can type your AWS account email address and password.

2. In the upper right corner of the console, choose your account name or number and then choose **My Account**.
3. On the right side of the page, next to the **Account Settings** section, choose **Edit**.
4. On the **Password** line choose **Edit** to change your password.
5. Choose a strong password. Although you can [set an account password policy for IAM users \(p. 75\)](#), that policy does not apply to your AWS account root user.

AWS requires that your password meet these conditions:

- have a minimum of 8 characters and a maximum of 128 characters
- include a minimum of three of the following mix of character types: uppercase, lowercase, numbers, and ! @ # \$ % ^ & * () <> [] {} | _+=- symbols
- not be identical to your AWS account name or email address

Note

AWS is rolling out improvements to the sign-in process. One of those improvements is to enforce a more secure password policy for your account. If your account has been upgraded, you are required to meet the password policy above. If your account has not yet been upgraded, then AWS does not enforce this policy, but highly recommends that you follow its guidelines for a more secure password.

To protect your password, it's important to follow these best practices:

- Change your password periodically and keep your password private, since anyone who knows your password may access your account.
- Use a different password on AWS than you use on other sites.
- Avoid passwords that are easy to guess. These include passwords such as `secret`, `password`, `amazon`, or `123456`. They also include things like a dictionary word, your name, email address, or other personal information that can easily be obtained.

Setting an Account Password Policy for IAM Users

You can set a password policy on your AWS account to specify complexity requirements and mandatory rotation periods for your IAM users' passwords.

You can use a password policy to do these things:

- Set a minimum password length.
- Require specific character types, including uppercase letters, lowercase letters, numbers, and non-alphanumeric characters. Be sure to remind your users that passwords are case sensitive.
- Allow all IAM users to change their own passwords.

Note

When you allow your IAM users to change their own passwords, IAM automatically allows them to view the password policy. IAM users need permission to view the account's password policy in order to create a password that complies with the policy.

- Require IAM users to change their password after a specified period of time (enable password expiration).
- Prevent IAM users from reusing previous passwords.
- Force IAM users to contact an account administrator when the user has allowed his or her password to expire.

Important

The password settings described here apply only to **passwords** assigned to IAM users and do not affect any access keys they might have. If a password expires, the user cannot sign-in to the AWS Management Console. However, if the user has valid access keys, then the user can still run any AWS CLI or Tools for Windows PowerShell commands or call any APIs through an application that the user's permissions allow.

When you create or change a password policy, most of the password policy settings are enforced the next time your users change their passwords, but some of the settings are enforced immediately. For example:

- When you set minimum length and character type requirements, the settings are enforced the next time your users change their passwords. Users are not forced to change their existing passwords, even if the existing passwords do not adhere to the updated password policy.
- When you set a password expiration period, the expiration period is enforced immediately. For example, when you set a password expiration period of 90 days, all IAM users that currently have an existing password that is more than 90 days old are forced to change their password at next sign-in.

For information about the permissions that you need in order to set a password policy, see [Permitting IAM Users to Change Their Own Passwords \(p. 82\)](#).

The IAM password policy does not apply to the AWS root account password.

The options currently available do not allow you to create what is commonly called a "lockout policy" that locks a user out of the account after a specified number of failed sign-in attempts. To get that kind of enhanced security, we recommend that you combine password policies together with multi-factor authentication (MFA). For more information about MFA, see [Using Multi-Factor Authentication \(MFA\) in AWS \(p. 91\)](#).

Topics

- [Password Policy Options \(p. 76\)](#)
- [Setting a Password Policy \(AWS Management Console\) \(p. 78\)](#)
- [Setting a Password Policy \(AWS CLI, Tools for Windows PowerShell, or AWS API\) \(p. 78\)](#)

Password Policy Options

The following list describes the options that are available when you configure a password policy for your account.

Minimum password length

You can specify the minimum number of characters allowed in an IAM user password. You can enter any number from 6 to 128.

Require at least one uppercase letter

You can require that IAM user passwords contain at least one uppercase character from the ISO basic Latin alphabet (A to Z).

Require at least one lowercase letter

You can require that IAM user passwords contain at least one lowercase character from the ISO basic Latin alphabet (a to z).

Require at least one number

You can require that IAM user passwords contain at least one numeric character (0 to 9).

Require at least one nonalphanumeric character

You can require that IAM user passwords contain at least one of the following nonalphanumeric characters:

! @ # \$ % ^ & * () _ + - = [] { } | '

Allow users to change their own password

You can permit all IAM users in your account to use the IAM console to change their own passwords, as described in [Permitting IAM Users to Change Their Own Passwords \(p. 82\)](#).

Alternatively, you can let only some users manage passwords, either for themselves or for others. To do so, you clear the **Allow users to change their own password** check box. For more information about using policies to limit who can manage passwords, see [Permitting IAM Users to Change Their Own Passwords \(p. 82\)](#).

Note

When you allow your IAM users to change their own passwords, IAM automatically allows them to view the password policy. IAM users need permission to view the account's password policy in order to create a password that complies with the policy.

Enable password expiration

You can set IAM user passwords to be valid for only the specified number of days. You specify the number of days that passwords remain valid after they are set. For example, when you enable password expiration and set the password expiration period to 90 days, an IAM user can use a password for up to 90 days. After 90 days, the password expires and the IAM user must set a new password before accessing the AWS Management Console. You can choose a password expiration period between 1 and 1095 days, inclusive.

Note

The AWS Management Console warns IAM users when they are within 15 days of password expiration. IAM users can change their password at any time (as long as they have been given permission to do so). When they set a new password, the rotation period for that password starts over. An IAM user can have only one valid password at a time.

Prevent password reuse

You can prevent IAM users from reusing a specified number of previous passwords. You can set the number of previous passwords from 1 to 24, inclusive.

Password expiration requires administrator reset

You can prevent IAM users from choosing a new password after their current password has expired. For example, if the password policy specifies a password expiration period, but an IAM user fails to choose a new password before the expiration period ends, the IAM user cannot set a new password.

In that case, the IAM user must request a password reset from an account administrator in order to regain access to the AWS Management Console. If you leave this check box cleared and an IAM user allows his or her password to expire, the user will be required to set a new password before accessing the AWS Management Console.

Warning

Before you enable this option, be sure that your AWS account has more than one user with administrative permissions (that is, permission to reset IAM user passwords) or that your administrators also have access keys that enable them to use the AWS CLI or Tools for Windows PowerShell separately from the AWS Management Console. When this option is enabled and one administrator's password expires, a second administrator is required to sign-in to the console to reset the expired password of the first administrator. However, if the administrator with the expired password has valid access keys then he or she can run an AWS CLI or Tools for Windows PowerShell command to reset his or her own password. The requirement for a second administrator applies only if a password expires and the first administrator has no access keys.

Setting a Password Policy (AWS Management Console)

You can use the AWS Management Console to create, change, or delete a password policy. As part of managing the password policy, you can let all users manage their own passwords.

To create or change a password 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, click **Account Settings**.
3. In the **Password Policy** section, select the options you want to apply to your password policy.
4. Click **Apply Password Policy**.

To delete a password 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, click **Account Settings**, and then in the **Password Policy** section, click **Delete Password Policy**.

Setting a Password Policy (AWS CLI, Tools for Windows PowerShell, or AWS API)

To manage an account password policy from the AWS CLI, Tools for Windows PowerShell, or AWS APIs, use the following commands:

To create or change a password policy

- AWS CLI: `aws iam update-account-password-policy`
- Tools for Windows PowerShell: `Update-IAMAccountPasswordPolicy`
- AWS API: `UpdateAccountPasswordPolicy`

To retrieve the password policy

- AWS CLI: `aws iam get-account-password-policy`
- Tools for Windows PowerShell: `Get-IAMAccountPasswordPolicy`
- AWS API: `GetAccountPasswordPolicy`

To delete a password policy

- AWS CLI: `aws iam delete-account-password-policy`
- Tools for Windows PowerShell: `Remove-IAMAccountPasswordPolicy`
- AWS API: `DeleteAccountPasswordPolicy`

Managing Passwords for IAM Users

IAM users who use the AWS Management Console to work with AWS resources must have a password in order to sign in. You can create, change, or delete a password for an IAM user in your AWS account.

After you have assigned a password to a user, the user can sign in to the AWS Management Console using the sign-in URL for your account, which looks like this:

`https://12-digit-AWS-account-ID or alias.signin.aws.amazon.com/console`

For more information about how IAM users sign in to the AWS Management Console, see [The IAM Console and Sign-in Page \(p. 52\)](#).

In addition to manually creating individual passwords for your IAM users, you can create a password policy that applies to all IAM user passwords in your AWS account.

You can use a password policy to do these things:

- Set a minimum password length.
- Require specific character types, including uppercase letters, lowercase letters, numbers, and non-alphanumeric characters. Be sure to remind your users that passwords are case sensitive.
- Allow all IAM users to change their own passwords.

Note

When you allow your IAM users to change their own passwords, IAM automatically allows them to view the password policy. IAM users need permission to view the account's password policy in order to create a password that complies with the policy.

- Require IAM users to change their password after a specified period of time (enable password expiration).
- Prevent IAM users from reusing previous passwords.
- Force IAM users to contact an account administrator when the user has allowed his or her password to expire.

For information about managing your account's password policy, see [Setting an Account Password Policy for IAM Users \(p. 75\)](#).

Even if your users have their own passwords, they still need permissions to access your AWS resources. By default, a user has no permissions. To give your users the permissions they need, you assign policies to them or to the groups they belong to. For information about creating users and groups, see [Identities \(Users, Groups, and Roles\) \(p. 59\)](#). For information about using policies to set permissions, see [Changing Permissions for an IAM User \(p. 71\)](#).

You can grant users permission to change their own passwords. For more information, see [Permitting IAM Users to Change Their Own Passwords \(p. 82\)](#). For information about how users access your account sign-in page, see [The IAM Console and Sign-in Page \(p. 52\)](#).

Topics

- [Creating, Changing, or Deleting an IAM User Password \(Console\) \(p. 80\)](#)
- [Creating, Changing, or Deleting an IAM User Password \(API, CLI, PowerShell\) \(p. 81\)](#)

Creating, Changing, or Deleting an IAM User Password (Console)

You can use the AWS Management Console to manage passwords for your IAM users.

When users leave your organization or no longer need AWS access, it is important to find the credentials that they were using and ensure that they are no longer operational. Ideally, you delete credentials if they are no longer needed. You can always recreate them at a later date if the need arises. At the very least you should change the credentials so that the former users no longer have access.

To add a password for an IAM user

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, choose **Users**.
3. Choose the name of the user whose password you want to create.
4. Choose the **Security credentials** tab, and then under **Sign-in credentials**, choose **Manage password** next to **Console password**.
5. In **Manage console access**, for **Console access** choose **Enable** if not already selected. If console access is disabled, then no password is needed.
6. For **Set password**, choose whether to have IAM generate a password or create a custom password:
 - To have IAM generate a password, choose **Autogenerated password**.
 - To create a custom password, choose **Custom password**, and type the password.

Note

The password that you create must meet the account's [password policy \(p. 75\)](#), if one is currently set.

7. To require the user to create a new password when signing in, choose **Require password reset**. Then choose **Apply**.

Important

If you select the **Require password reset** option, make sure that the user has permission to change his or her password. For more information, see [Permitting IAM Users to Change Their Own Passwords \(p. 82\)](#).

8. If you choose the option to autogenerated a password, choose **Show** in the **New password** dialog box. This lets you view the password so you can share it with the user.

Important

For security reasons, you cannot access the password after completing this step, but you can create a new password at any time.

To change the password for an IAM user

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, choose **Users**.
3. Choose the name of the user whose password you want to change.
4. Choose the **Security credentials** tab, and then under **Sign-in credentials**, choose **Manage password** next to **Console password**.
5. In **Manage console access**, for **Console access** choose **Enable** if not already selected. If console access is disabled, then no password is needed.
6. For **Set password**, choose whether to have IAM generate a password or create a custom password:
 - To have IAM generate a password, choose **Autogenerated password**.
 - To create a custom password, choose **Custom password**, and type the password.

Note

The password that you create must meet the account's [password policy \(p. 75\)](#), if one is currently set.

7. To require the user to create a new password when signing in, choose **Require password reset**. Then choose **Apply**.

Important

If you select the **Require password reset** option, make sure that the user has permission to change his or her password. For more information, see [Permitting IAM Users to Change Their Own Passwords \(p. 82\)](#).

8. If you choose the option to autogenerated a password, choose **Show** in the **New password** dialog box. This lets you view the password so you can share it with the user.

Important

For security reasons, you cannot access the password after completing this step, but you can create a new password at any time.

To delete an IAM user's password

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, choose **Users**.
3. Choose the name of the user whose password you want to delete.
4. Choose the **Security credentials** tab, and then under **Sign-in credentials**, choose **Manage password** next to **Console password**.
5. For **Console access**, choose **Disable**, and then choose **Apply**.

Important

When you remove a user's password, the user can no longer sign in to the AWS Management Console. If the user has active access keys, they continue to function and allow access through the AWS CLI, Tools for Windows PowerShell, or AWS API function calls.

[Creating, Changing, or Deleting an IAM User Password \(API, CLI, PowerShell\)](#)

To manage passwords for IAM users, use the following commands:

To create a password

- AWS CLI: [aws iam create-login-profile](#)
- Tools for Windows PowerShell: [New-IAMLoginProfile](#)
- AWS API: [CreateLoginProfile](#)

To determine whether a user has a password

- AWS CLI: [aws iam get-login-profile](#)
- Tools for Windows PowerShell: [Get-IAMLoginProfile](#)
- AWS API: [GetLoginProfile](#)

To determine when a user's password was last used

- AWS CLI: [aws iam get-user](#)
- Tools for Windows PowerShell: [Get-IAMUser](#)
- AWS API: [GetUser](#)

To change a user's password

- AWS CLI: [aws iam update-login-profile](#)
- Tools for Windows PowerShell: [Update-IAMLoginProfile](#)
- AWS API: [UpdateLoginProfile](#)

To delete a user's password

- AWS CLI: [aws iam delete-login-profile](#)
- Tools for Windows PowerShell: [Remove-IAMLoginProfile](#)
- AWS API: [DeleteLoginProfile](#)

Note

You can use the AWS CLI, Tools for Windows PowerShell, or AWS API to delete a user from your AWS account. However, you must first delete the password as a separate step in the process of removing the user. For more information, see [Deleting an IAM User \(AWS CLI and Tools for Windows PowerShell\) \(p. 70\)](#).

Permitting IAM Users to Change Their Own Passwords

You can grant IAM users the permission to change their own passwords for signing in to the AWS Management Console. You can do this in one of two ways:

- [Allow all IAM users in the account to change their own passwords \(p. 82\).](#)
- [Allow only selected IAM users to change their own passwords \(p. 82\).](#) In this scenario, you disable the option for all users to change their own passwords and you use an IAM policy to grant permissions to only some users to change their own passwords and optionally other credentials like their own access keys.

Important

We recommend that you [set a password policy \(p. 75\)](#) so that users create strong passwords.

To allow all IAM users change their own passwords

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, click **Account Settings**.
3. In the **Password Policy** section, select **Allow users to change their own password**, and then click **Apply Password Policy**.
4. Point users to the following instructions that show how they can change their passwords: [How IAM Users Change Their Own Password \(p. 83\)](#).

For information about the AWS CLI, Tools for Windows PowerShell, and API commands that you can use to change the account's password policy (which includes letting all users change their own passwords), see [Setting a Password Policy \(AWS CLI, Tools for Windows PowerShell, or AWS API\) \(p. 78\)](#).

To allow selected IAM users change their own passwords

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, click **Account Settings**.

3. In the **Account Settings** section, make sure that **Allow users to change their own password** is not selected. If this check box is selected, all users can change their own passwords. (See the previous procedure.)
4. Create the users who should be able to change their own password, if they do not already exist. For details, see [Creating an IAM User in Your AWS Account \(p. 63\)](#).
5. Create an IAM group for the users who should be allowed to change their passwords, and then add the users from the previous step to the group. For details, see [Creating Your First IAM Admin User and Group \(p. 17\)](#) and [Managing IAM Groups \(p. 131\)](#).

This step is optional, but it's a best practice to use groups to manage permissions so that you can add and remove users and change the permissions for the group as a whole.

6. Assign the following policy to the group. For details, see [Managing IAM Policies \(p. 316\)](#).

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "iam:GetAccountPasswordPolicy",  
            "Resource": "*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": "iam:ChangePassword",  
            "Resource": "arn:aws:iam::account-id-without-hyphens:user/${aws:username}"  
        }  
    ]  
}
```

This policy grants access to the [ChangePassword](#) action, which lets users change only their own passwords from the console, the AWS CLI, Tools for Windows PowerShell, or the API. It also grants access to the [GetAccountPasswordPolicy](#) action, which lets the user view the current password policy; this permission is required so that the user can display the **Change Password** page in the console. The user must be able to read the current password policy to ensure the changed password meets the requirements of the policy.

7. Point users to the following instructions that show how they can change their passwords: [How IAM Users Change Their Own Password \(p. 83\)](#).

For More Information

For more information on managing credentials, see the following topics:

- [Permitting IAM Users to Change Their Own Passwords \(p. 82\)](#)
- [Managing Passwords \(p. 74\)](#)
- [Setting an Account Password Policy for IAM Users \(p. 75\)](#)
- [Managing IAM Policies \(p. 316\)](#)
- [How IAM Users Change Their Own Password \(p. 83\)](#)

How IAM Users Change Their Own Password

If IAM users have been granted permission to change their own passwords, they can use a special page in the AWS Management Console to do this. They can also use the command line interface or the IAM API.

For information about the permissions that users need in order to change their own passwords, see [Permitting IAM Users to Change Their Own Passwords \(p. 82\)](#).

Topics

- [How IAM Users Change Their Own Password \(AWS Management Console\) \(p. 84\)](#)
- [How IAM Users Change Their Own Password \(AWS CLI, Tools for Windows PowerShell, or AWS API\) \(p. 84\)](#)

How IAM Users Change Their Own Password (AWS Management Console)

The following procedure describes how IAM users can use the AWS Management Console to change their own password.

To use the console to change your own password as an IAM user

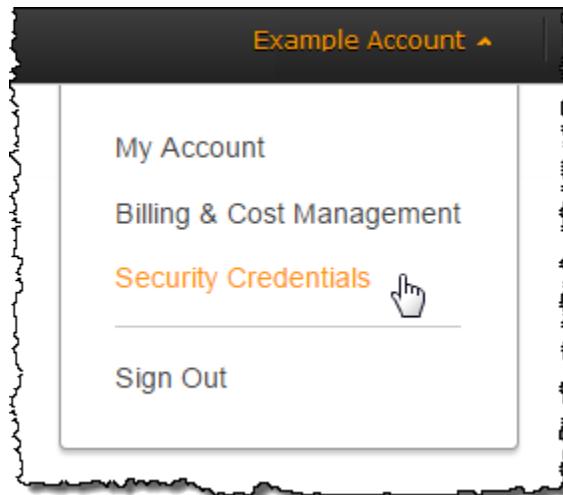
1. Use your AWS account ID or account alias, your IAM user name, and your password to sign in to the [IAM console](#).

Note

For your convenience, the AWS sign-in page uses a browser cookie to remember your IAM user name and account information. If you previously signed in as a different user, choose **Sign in to a different account** near the bottom of the page to return to the main sign-in page. From there, you can type your AWS account ID or account alias to be redirected to the IAM user sign-in page for your account.

To get your AWS account ID, contact your administrator.

2. In the navigation bar on the upper right, choose your user name, and then choose **Security Credentials**.



3. For **Current password**, type your current password. Type a new password in the **New password** and **Confirm new password** boxes. Then click **Submit**.

Note

If the account has a password policy, the new password must meet the requirements of that policy. For more information, see [Setting an Account Password Policy for IAM Users \(p. 75\)](#).

How IAM Users Change Their Own Password (AWS CLI, Tools for Windows PowerShell, or AWS API)

The following procedure describes how IAM users can use the AWS CLI, Tools for Windows PowerShell, or AWS API to change their own password.

To change your own IAM password, use the following commands

- AWS CLI: `aws iam change-password`
- Tools for Windows PowerShell: [Edit-IAMPassword](#)
- AWS API: `ChangePassword`

Managing Access Keys for IAM Users

 Follow us on Twitter

Note

If you found this topic because you are trying to configure the Product Advertising API to sell Amazon products on your website, see these topics:

- [Getting Started with the Product Advertising API](#)
- [Getting Started as a Product Advertising API Developer](#)

Users need their own access keys to make programmatic calls to AWS from the [AWS Command Line Interface](#) (AWS CLI), [Tools for Windows PowerShell](#), the [AWS SDKs](#), or direct HTTP calls using the APIs for individual AWS services. To fill this need, you can create, modify, view, or rotate access keys (access key IDs and secret access keys) for IAM users.

When you create an access key, IAM returns the access key ID and secret access key. You should save these in a secure location and give them to the user.

Important

To ensure the security of your AWS account, the secret access key is accessible only at the time you create it. If a secret access key is lost, you must delete the access key for the associated user and create a new key. For more details, see [Resetting Your Lost or Forgotten Passwords or Access Keys \(p. 90\)](#).

By default, when you create an access key, its status is `Active`, which means the user can use the access key for AWS CLI, Tools for Windows PowerShell, and API calls. Each user can have two active access keys, which is useful when you must rotate the user's access keys. You can disable a user's access key, which means it can't be used for API calls. You might do this while you're rotating keys or to revoke API access for a user.

You can delete an access key at any time. However, when you delete an access key, it's gone forever and cannot be retrieved. (You can always create new keys.)

You can give your users permission to list, rotate, and manage their own keys. For more information, see [Allow Users to Manage Their Own Passwords, Access Keys, and SSH Keys \(p. 377\)](#).

For more information about the credentials used with AWS and IAM, see [Temporary Security Credentials \(p. 231\)](#), and [Types of Security Credentials](#) in the *Amazon Web Services General Reference*.

Topics

- [Creating, Modifying, and Viewing Access Keys \(Console\) \(p. 85\)](#)
- [Creating, Modifying, and Viewing Access Keys \(API, CLI, PowerShell\) \(p. 87\)](#)
- [Rotating Access Keys \(p. 88\)](#)

Creating, Modifying, and Viewing Access Keys (Console)

You can use the AWS Management Console to manage the access keys of IAM users.

To list the access key IDs for multiple users

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, choose **Users**.
3. If necessary, add the **Access key ID** column to the users table by completing the following steps:
 - a. Above the table on the far right, choose the settings icon ().
 - b. In **Manage Columns**, select **Access key ID**.
 - c. Choose **Close** to return to the list of users.
4. The **Access key ID** column includes the access key IDs. You can use this information to view and copy the access keys for users with one or two access keys. The column also shows whether the access key is **(Active)** or **(Inactive)**. The column displays **None** for users with no access key.

Note

Only the user's access key ID and status is visible. The secret access key can only be retrieved when the key is created.

To find which user owns a specific access key

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, choose **Users**.
3. In the search box, type or paste the access key ID of the user you want to find.
4. If necessary, add the **Access key ID** column to the users table by completing the following steps:
 - a. Above the table on the far right, choose the settings icon ().
 - b. In **Manage Columns**, select **Access key ID**.
 - c. Choose **Close** to return to the list of users and confirm that the filtered user owns the specified access key.

To list a user's access keys

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, choose **Users**.
3. Choose the name of the intended user, and then choose the **Security Credentials** tab. The user's access keys and the status of each key is displayed.

Note

Only the user's access key ID is visible. The secret access key can only be retrieved when the key is created.

To create, modify, or delete a user's access keys

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, choose **Users**.
3. Choose the name of the desired user, and then choose the **Security Credentials** tab.
4. If needed, expand the **Access Keys** section and do any of the following:

- To create an access key, choose **Create Access Key**. Then choose **Download Credentials** to save the access key ID and secret access key to a CSV file on your computer. Store the file in a secure location. You will not have access to the secret access key again after this dialog box closes. After you have downloaded the CSV file, choose **Close**.
- To disable an active access key, choose **Make Inactive**.
- To reenable an inactive access key, choose **Make Active**.
- To delete an access key, choose **Delete** and then choose **Delete** to confirm.

Creating, Modifying, and Viewing Access Keys (API, CLI, PowerShell)

To manage a user's access keys from the AWS CLI, Tools for Windows PowerShell, or the AWS API, use the following commands:

To create an access key

- AWS CLI: `aws iam create-access-key`
- Tools for Windows PowerShell: [New-IAMAccessKey](#)
- AWS API: [CreateAccessKey](#)

To disable or reenable an access key

- AWS CLI: `aws iam update-access-key`
- Tools for Windows PowerShell: [Update-IAMAccessKey](#)
- AWS API: [UpdateAccessKey](#)

To list a user's access keys

- AWS CLI: `aws iam list-access-keys`
- Tools for Windows PowerShell: [Get-IAMAccessKey](#)
- AWS API: [ListAccessKeys](#)

To determine when an access key was most recently used

- AWS CLI: `aws iam get-access-key-last-used`
- Tools for Windows PowerShell: [Get-IAMAccessKeyLastUsed](#)
- AWS API: [GetAccessKeyLastUsed](#)

To delete an access key

- AWS CLI: `aws iam delete-access-key`
- Tools for Windows PowerShell: [Remove-IAMAccessKey](#)
- AWS API: [DeleteAccessKey](#)

Rotating Access Keys

As a security best practice, we recommend that you, an administrator, regularly rotate (change) the access keys for IAM users in your account. If your users have the necessary permissions, they can rotate their own access keys. For information about how to give your users permissions to rotate their own access keys, see [Allow Users to Manage Their Own Passwords, Access Keys, and SSH Keys \(p. 377\)](#).

You can also apply a password policy to your account to require that all of your IAM users periodically rotate their passwords. You can choose how often they must do so. For more information, see [Setting an Account Password Policy for IAM Users \(p. 75\)](#).

Important

If you use the AWS account root user credentials, we recommend that you also regularly rotate them. The account password policy does not apply to the root user credentials. IAM users cannot manage credentials for the AWS account root user, so you must use the root user credentials (not a user's) to change the root user credentials. Note that we recommend against using the root user for everyday work in AWS.

To determine when access keys needs rotating (console)

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, choose **Users**.
3. If necessary, add the **Access key age** column to the users table by completing the following steps:
 - a. Above the table on the far right, choose the settings icon ().
 - b. In **Manage Columns**, select **Access key age**.
 - c. Choose **Close** to return to the list of users.
4. The **Access key age** column shows the number of days since the oldest active access key was created. You can use this information to find users with access keys that need rotating. The column displays **None** for users with no access key.

To rotate access keys without interrupting your applications (console)

The following steps describe the general process for rotating an access key without interrupting your applications. These steps show the AWS CLI, Tools for Windows PowerShell and AWS API commands for rotating access keys. You can also perform these tasks using the console; for details, see [Creating, Modifying, and Viewing Access Keys \(Console\) \(p. 85\)](#), in the section above.

1. While the first access key is still active, create a second access key, which is active by default. At this point, the user has two active access keys.
 - a. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
 - b. In the navigation pane, choose **Users**.
 - c. Choose the name of the intended user, and then choose the **Security Credentials** tab.
 - d. Choose **Create Access Key** and then choose **Download Credentials** to save the access key ID and secret access key to a .csv file on your computer. Store the file in a secure location. You will not have access to the secret access key again after this closes. After you have downloaded the .csv file, choose **Close**.
2. Update all applications and tools to use the new access key.
3. Determine whether the first access key is still in use by reviewing the **Last used** column for the oldest access key. One approach is to wait several days and then check the old access key for any use before proceeding.

4. Even if the **Last used** column value indicates that the old key has never been used, we recommend that you do not immediately delete the first access key. Instead, choose **Make inactive** to deactivate the first access key.
5. Use only the new access key to confirm that your applications are working. Any applications and tools that still use the original access key will stop working at this point because they no longer have access to AWS resources. If you find such an application or tool, you can choose **Make active** to reenable the first access key. Then return to [Step 3 \(p. 88\)](#) and update this application to use the new key.
6. After you wait some period of time to ensure that all applications and tools have been updated, you can delete the first access key:
 - a. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
 - b. In the navigation pane, choose **Users**.
 - c. Choose the name of the intended user, and then choose the **Security Credentials** tab.
 - d. Choose **Create Access Key**, choose **Delete**, and then choose **Delete** to confirm.

To rotate access keys without interrupting your applications (API, CLI, PowerShell)

1. While the first access key is still active, create a second access key, which is active by default. At this point, the user has two active access keys.
 - AWS CLI: `aws iam create-access-key`
 - Tools for Windows PowerShell: `New-IAMAccessKey`
 - AWS API: `CreateAccessKey`
 2. Update all applications and tools to use the new access key.
 3. Determine whether the first access key is still in use:
 - AWS CLI: `aws iam get-access-key-last-used`
 - Tools for Windows PowerShell: `Get-IAMAccessKeyLastUsed`
 - AWS API: `GetAccessKeyLastUsed`
- One approach is to wait several days and then check the old access key for any use before proceeding.
4. Even if step [Step 3](#) indicates no use of the old key, we recommend that you do not immediately delete the first access key. Instead, change the state of the first access key to **Inactive**.
 - AWS CLI: `aws iam update-access-key`
 - Tools for Windows PowerShell: `Update-IAMAccessKey`
 - AWS API: `UpdateAccessKey`
 5. Use only the new access key to confirm that your applications are working. Any applications and tools that still use the original access key will stop working at this point because they no longer have access to AWS resources. If you find such an application or tool, you can switch its state back to **Active** to reenable the first access key. Then return to step [Step 2](#) and update this application to use the new key.
 6. After you wait some period of time to ensure that all applications and tools have been updated, you can delete the first access key.
 - AWS CLI: `aws iam delete-access-key`
 - Tools for Windows PowerShell: `Remove-IAMAccessKey`
 - AWS API: `DeleteAccessKey`

For more information, see the following:

- [How to Rotate Access Keys for IAM Users](#). This entry on the *AWS Security Blog* provides more information on key rotation.
- [IAM Best Practices \(p. 43\)](#). This page provides general recommendations for helping to secure your AWS resources.

Resetting Your Lost or Forgotten Passwords or Access Keys

If you lose or forget your passwords or access keys, you *cannot* retrieve them from IAM. Instead, you can reset them using the following methods:

- **AWS account root user password** – If you forget your root user password, you can reset the password from the AWS Management Console. For details, see [the section called "Resetting a Lost or Forgotten Root User Password" \(p. 90\)](#) later in this topic.
- **AWS account access keys** – If you forget your account access keys, you can create new access keys without disabling the existing access keys. If you are not using the existing keys, you can delete those. For details, see [Creating Access Keys for the Root User \(p. 261\)](#) and [Deleting Access Keys from the Root User \(p. 262\)](#).
- **IAM user password** – If you are an IAM user and you forget your password, you must ask your administrator to reset your password. To learn how an administrator can manage your password, see [Managing Passwords for IAM Users \(p. 79\)](#).
- **IAM user access keys** – If you are an IAM user and you forget your access keys, you will need new access keys. If you have permission to create your own access keys, you can find instructions for creating a new one at [Creating, Modifying, and Viewing Access Keys \(Console\) \(p. 85\)](#). If you do not have the required permissions, you must ask your administrator to create new access keys. If you are still using your old keys, ask your administrator not to delete the old keys. To learn how an administrator can manage your access keys, see [Managing Access Keys for IAM Users \(p. 85\)](#).

You should follow the [AWS best practice \(p. 47\)](#) of periodically changing your password and AWS access keys. In AWS, you change access keys by *rotating* them. This means that you create a new one, configure your applications to use the new key, and then delete the old one. You are allowed to have two access key pairs active at the same time for just this reason. For more information, see [Rotating Access Keys \(p. 88\)](#).

Resetting a Lost or Forgotten Root User Password

When you first created your AWS account, you provided an email address and password. These are your AWS account root user credentials. If you forget your root user password, you can reset the password from the AWS Management Console.

To reset your root user password:

1. Use your AWS account email address to begin signing in to the [AWS Management Console](#) as the root user.

Note

If you are signed in to the [AWS Management Console](#) with *IAM user* credentials, then you must sign out before you can reset the root user password. If you see the account-specific IAM user sign-in page, choose **Sign-in using root account credentials** near the bottom of the page. If necessary, provide your account email address to access the **Root user sign in** page.

2. Choose **Forgot your password?**.

3. Provide the email address that you used to create the account. Then provide the CAPTCHA text and choose **Continue**.
4. Check the email that is associated with your AWS account for a message from Amazon Web Services. The email will come from an address ending in @amazon.com or @aws.amazon.com. Follow the directions in the email. If you don't see the email in your account, check your spam folder. If you no longer have access to the email, see [I need to access an old account \(p. 383\)](#).

Using Multi-Factor Authentication (MFA) in AWS

 [Follow us on Twitter](#)

For increased security, we recommend that you configure multi-factor authentication (MFA) to help protect your AWS resources. MFA adds extra security because it requires users to type a unique authentication code from an approved authentication device or SMS text message when they access AWS websites or services.

- **Security token-based.** This type of MFA requires you to assign an MFA device (hardware or virtual) to the IAM user or the AWS account root user. A virtual device is a software application running on a phone or other mobile device that emulates a physical device. Either way, the device generates a six-digit numeric code based upon a time-synchronized one-time password algorithm. The user must type a valid code from the device on a second webpage during sign-in. Each MFA device assigned to a user must be unique; a user cannot type a code from another user's device to authenticate. For more information about enabling security token-based MFA, see [Enabling a Hardware MFA Device \(Console\) \(p. 96\)](#) and [Enabling a Virtual Multi-factor Authentication \(MFA\) Device \(p. 93\)](#).
- **SMS text message-based.** This type of MFA requires you to configure the IAM user with the phone number of the user's SMS-compatible mobile device. When the user signs in, AWS sends a six-digit numeric code by SMS text message to the user's mobile device. The user is required to type that code on a second webpage during sign-in. Note that SMS-based MFA is available only for IAM users. You cannot use this type of MFA with the AWS account root user. For more information about enabling SMS text messaging-based MFA, see [PREVIEW - Enabling SMS Text Message MFA Devices \(p. 99\)](#).

Note

AWS is no longer accepting new participants for the SMS MFA preview. We encourage you to use MFA on your AWS account through either a [hardware-based \(p. 96\)](#) or [virtual \(software-based\) \(p. 93\)](#) MFA token device. If your account is already participating in the SMS MFA Preview program, you can continue using this feature.

No matter how the user receives the six-digit numeric MFA code, the user types it on a second page of the sign-in process for the AWS Management Console. In some cases, a user may work with the AWS STS API or CLI instead of the console. In that case, you can pass hardware or virtual MFA device codes as parameters to STS API calls to get temporary credentials.

Note

Currently, you can use SMS-based MFA only with AWS Management Console. You cannot use SMS-based MFA with the API or AWS CLI.

This section shows you how to configure MFA for your users and set them up to use token devices or SMS text messages. It also describes how to synchronize and deactivate existing token devices, and what to do when a device is lost or stops working.

Note

When you enable MFA for the AWS account root user, it affects only the root user credentials. IAM users in the account are distinct identities with their own credentials, and each identity has its own MFA configuration.

For answers to commonly asked questions about AWS MFA, go to the [AWS Multi-Factor Authentication FAQs](#).

Topics

- [Enabling MFA Devices \(p. 92\)](#)
- [Checking MFA Status \(p. 101\)](#)
- [Resynchronize MFA Devices \(p. 102\)](#)
- [Deactivating MFA Devices \(p. 104\)](#)
- [What If an MFA Device Is Lost or Stops Working? \(p. 106\)](#)
- [Configuring MFA-Protected API Access \(p. 107\)](#)
- [Sample Policies with MFA Conditions \(p. 112\)](#)
- [Sample Code: Requesting Credentials with Multi-factor Authentication \(p. 115\)](#)

Enabling MFA Devices

The following overview procedure describes how to set up and use MFA and provides links to related information.

1. *Get an MFA token device or an SMS-compatible mobile device such as one of the following.* You can enable only one MFA device per AWS account root user or IAM user, and the device can only be used by the specified user.
 - A hardware-based token device, such as one of the AWS supported hardware token devices discussed on the [Multi-Factor Authentication](#) page.
 - A virtual token device, which is a software application that is compliant with [RFC 6238, a standards-based TOTP \(time-based one-time password\) algorithm](#). You can install the application on a mobile device, such as a tablet or smartphone. For a list of a few supported apps that you can use as virtual MFA devices, see the [Virtual MFA Applications](#) section of the [Multi-Factor Authentication](#) page.
 - A mobile phone that can receive standard SMS text messages.

Note

If you choose to use SMS-based MFA, text messaging charges from your mobile device's carrier may apply.

SMS-based MFA are only available for IAM users and cannot be used for the root user.

2. *Enable the MFA device.* Enabling a device has two steps. First, you create an MFA device entity in IAM. Then you associate the MFA device entity with the IAM user. You can perform these tasks in the AWS Management Console, the AWS CLI, Tools for Windows PowerShell or the IAM API.

For information about enabling each type of MFA device, see the following topics:

- Physical MFA device: [Enabling a Hardware MFA Device \(Console\) \(p. 96\)](#)
- Virtual MFA device: [Enabling a Virtual Multi-factor Authentication \(MFA\) Device \(p. 93\)](#)
- SMS MFA device: [PREVIEW - Enabling SMS Text Message MFA Devices \(p. 99\)](#)

3. *Use the MFA device when you log in to or access AWS resources.* Note the following:

- To access an AWS website, you need an MFA code from the device in addition to the usual user name and password. If AWS determines that the IAM user you sign in as is MFA-enabled with SMS, then it automatically sends the MFA code to the configured phone number.
- To access MFA-protected API operations, you need an MFA code, the identifier for the MFA device (the device serial number of a physical device or the ARN of a virtual or SMS device defined in AWS), and the usual access key ID and secret access key.

Note

During the public preview of SMS MFA, you can authenticate with your SMS device only in the AWS Management Console. You cannot pass the MFA information for an SMS MFA device to AWS STS APIs to request temporary credentials.

For more information, see [Using MFA Devices With Your IAM Sign-in Page \(p. 56\)](#)

Topics

- [Enabling a Virtual Multi-factor Authentication \(MFA\) Device \(p. 93\)](#)
- [Enabling a Hardware MFA Device \(Console\) \(p. 96\)](#)
- [PREVIEW - Enabling SMS Text Message MFA Devices \(p. 99\)](#)
- [Enable and manage virtual MFA devices \(AWS CLI, Tools for Windows PowerShell, or AWS API\) \(p. 100\)](#)

Enabling a Virtual Multi-factor Authentication (MFA) Device

A virtual MFA device uses a software application to generate a six-digit authentication code that is compatible with the time-based one-time password (TOTP) standard, as described in [RFC 6238](#). The app can run on mobile hardware devices, including smartphones. With most virtual MFA apps, you can host more than one virtual MFA device, which makes them more convenient than hardware MFA devices. However, be aware that a virtual MFA might be run on a less secure device such as a smartphone. Consequently, a virtual MFA might not provide the same level of security as a hardware MFA device.

You can enable only one MFA device per AWS account root user or IAM user, and the device can only be used by the specified user. Keep in mind that although some virtual MFA software apps appear to support multiple accounts, each account you add represents a single virtual MFA device. In addition, that one virtual device can still associate with only one user.

For a list of virtual MFA apps that you can use on smartphones and tablets (including Google Android, Apple iPhone and iPad, and Windows Phone), go to the [Virtual MFA Applications](#) section at <http://aws.amazon.com/iam/details/mfa/>. Note that AWS requires a virtual MFA app that produces a six-digit OTP.

Use the following steps to enable and manage MFA devices from the AWS Management Console. To enable and manage MFA devices at the command line, or to use the API, see [Enable and manage virtual MFA devices \(AWS CLI, Tools for Windows PowerShell, or AWS API\) \(p. 100\)](#).

Important

We recommend that when you configure a virtual MFA device to use with AWS that you save a copy of the QR code or the secret key ***in a secure place***. That way, if you lose the phone or have to reinstall the MFA software app for any reason, you can reconfigure the app to use the same virtual MFA. This avoids the need to create a new virtual MFA in AWS for the user or root user.

Enable a Virtual MFA Device for an IAM User (AWS Management Console)

You can use IAM in the AWS Management Console to enable a virtual MFA device for an IAM user in your account.

Note

You must have physical access to the hardware that will host the user's virtual MFA device in order to configure MFA. For example, you might configure MFA for a user who will use a virtual MFA device running on a smartphone. In that case, you must have the smartphone available in order to finish the wizard. Because of this, you might want to let users configure and manage their own virtual MFA devices. In that case you must grant users the permissions to perform the necessary IAM actions. For more information and for an example of an IAM policy that grants these permissions, see [Allow Users to Manage Only Their Own Virtual MFA Devices \(p. 380\)](#).

To enable a virtual MFA device for a user

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, choose **Users**.
3. In the **User Name** list, choose the name of the intended MFA user.
4. Choose the **Security credentials** tab. Next to **Assigned MFA device**, choose the edit icon ().

5. In the **Manage MFA Device** wizard, choose **A virtual MFA device**, and then choose **Next Step**.

IAM generates and displays configuration information for the virtual MFA device, including a QR code graphic. The graphic is a representation of the 'secret configuration key' that is available for manual entry on devices that do not support QR codes.

6. Open your virtual MFA app. (For a list of apps that you can use for hosting virtual MFA devices, see [Virtual MFA Applications](#).) If the virtual MFA app supports multiple accounts (multiple virtual MFA devices), choose the option to create a new account (a new virtual MFA device).
7. Determine whether the MFA app supports QR codes, and then do one of the following:
 - Use the app to scan the QR code. For example, you might choose the camera icon or choose an option similar to **Scan code**, and then use the device's camera to scan the code.
 - In the **Manage MFA Device** wizard, choose **Show secret key for manual configuration**, and then type the secret configuration key into your MFA app.

When you are finished, the virtual MFA device starts generating one-time passwords.

8. In the **Manage MFA Device** wizard, in the **Authentication Code 1** box, type the one-time password that currently appears in the virtual MFA device. Wait up to 30 seconds for the device to generate a new one-time password. Then type the second one-time password into the **Authentication Code 2** box. Choose **Active Virtual MFA**.

Important

Submit your request immediately after generating the codes. If you generate the codes and then wait too long to submit the request, the MFA device successfully associates with the user but the MFA device is out of sync. This happens because time-based one-time passwords (TOTP) expire after a short period of time. If this happens, you can [resync the device](#) (p. 102).

The virtual MFA device is now ready for use with AWS. For information about using MFA with the AWS Management Console, see [Using MFA Devices With Your IAM Sign-in Page \(p. 56\)](#).

Enable a Virtual MFA Device for Your AWS Account Root User (Console)

You can use IAM in the AWS Management Console to configure and enable a virtual MFA device for your root user. To manage MFA devices for the AWS account, you must be signed in to AWS using your root user credentials. You cannot manage MFA devices for the root user using other credentials.

If your MFA device is lost, stolen, or not working, you can still sign in using alternative factors of authentication. To do this, you must verify your identity using the email and phone that are registered with your account. This means that if you can't sign in with your MFA device, you can sign in by verifying your identity using the email and phone that are registered with your account. Before you enable MFA for your root user, review your account settings and contact information to make sure that you have access to the email and phone number. To learn about signing in using alternative factors of authentication, see [What If an MFA Device Is Lost or Stops Working? \(p. 106\)](#). To disable this feature, contact [AWS Support](#).

Note

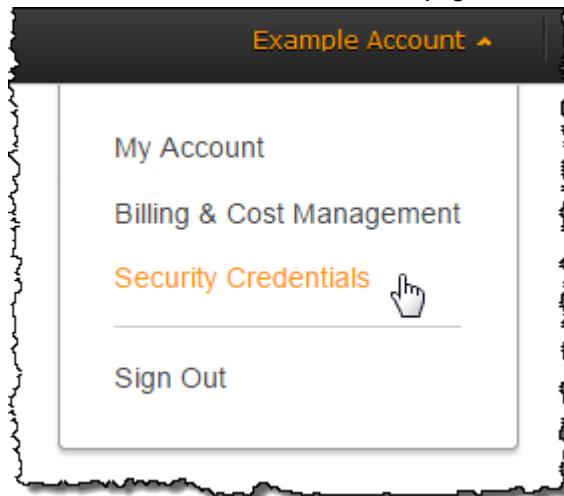
If you are using an AWS account created after September 14, 2017, you might see differences in the following console pages: **Sign in with authentication device** and **Troubleshoot your authentication device**. However, the same features are provided. In either case, if you cannot verify your account email address and phone number using alternative factors of authentication, contact [AWS Support](#) to deactivate your MFA setting.

To configure and enable a virtual MFA device for use with your root user

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.

2. Do one of the following:

- **Option 1:** Choose **Dashboard**, and under **Security Status**, expand **Activate MFA on your root user**.
- **Option 2:** On the right side of the navigation bar, choose your account name, and choose **Security Credentials**. If necessary, choose **Continue to Security Credentials**. Then expand the **Multi-Factor Authentication (MFA)** section on the page.



3. Choose **Manage MFA** or **Activate MFA**, depending on which option you chose in the preceding step.
4. In the wizard, choose **A virtual MFA device** and then choose **Next Step**.
5. Confirm that a virtual MFA app is installed on the device, and then choose **Next Step**. IAM generates and displays configuration information for the virtual MFA device, including a QR code graphic. The graphic is a representation of the secret configuration key that is available for manual entry on devices that do not support QR codes.
6. With the **Manage MFA Device** wizard still open, open the virtual MFA app on the device.
7. If the virtual MFA software supports multiple accounts (multiple virtual MFA devices), then choose the option to create a new account (a new virtual device).
8. The easiest way to configure the app is to use the app to scan the QR code. If you cannot scan the code, you can type the configuration information manually.
 - To use the QR code to configure the virtual MFA device, follow the app instructions for scanning the code. For example, you might need to tap the camera icon or tap a command like **Scan account barcode**, and then use the device's camera to scan the QR code.
 - If you cannot scan the code, type the configuration information manually by typing the **Secret Configuration Key** value into the app. For example, to do this in the AWS Virtual MFA app, choose **Manually add account**, and then type the secret configuration key and choose **Create**.

Important

Make a secure backup of the QR code or secret configuration key, or make sure that you enable multiple virtual MFA devices for your account. A virtual MFA device might become unavailable, for example, if you lose the smartphone where the virtual MFA device is hosted). If that happens, you will not be able to sign in to your account and you will have to contact customer service to remove MFA protection for the account.

Note

The QR code and secret configuration key generated by IAM are tied to your AWS account and cannot be used with a different account. They can, however, be reused to configure a new MFA device for your account in case you lose access to the original MFA device.

The device starts generating six-digit numbers.

9. In the **Manage MFA Device** wizard, in the **Authentication Code 1** box, type the six-digit number that's currently displayed by the MFA device. Wait up to 30 seconds for the device to generate a new number, and then type the new six-digit number into the **Authentication Code 2** box.

Important

Submit your request immediately after generating the codes. If you generate the codes and then wait too long to submit the request, the MFA device successfully associates with the user but the MFA device is out of sync. This happens because time-based one-time passwords (TOTP) expire after a short period of time. If this happens, you can [resync the device \(p. 102\)](#).

10. Choose **Next Step**, and then choose **Finish**.

The device is ready for use with AWS. For information about using MFA with the AWS Management Console, see [Using MFA Devices With Your IAM Sign-in Page \(p. 56\)](#).

Replace or "Rotate" a Virtual MFA Device

You can have only one virtual MFA device assigned to a user at a time. If the user loses a device or needs to replace it for any reason, you must first deactivate the old device. Then you can add the new device for the user.

- To deactivate the device currently associated with a user, see [Deactivating MFA Devices \(p. 104\)](#).
- To add a replacement virtual MFA device for an IAM user, follow the steps in the procedure [Enable a Virtual MFA Device for an IAM User \(AWS Management Console\) \(p. 93\)](#) above.
- To add a replacement virtual MFA device for the AWS account root user, follow the steps in the procedure [Enable a Virtual MFA Device for Your AWS Account Root User \(Console\) \(p. 94\)](#) earlier in this topic.

Enabling a Hardware MFA Device (Console)

Topics

- [Enable a Hardware MFA Device for an IAM User \(Console\) \(p. 96\)](#)
- [Enable a Hardware MFA Device for the AWS Account Root User \(Console\) \(p. 97\)](#)
- [Replace or "Rotate" a Physical MFA Device \(p. 99\)](#)

You can enable a hardware MFA device from the AWS Management Console, the command line, or the IAM API. The following procedure shows you how to use the AWS Management Console to enable the device for a user under your AWS account. To enable an MFA device for your AWS account root user, see [Enable a Hardware MFA Device for the AWS Account Root User \(Console\) \(p. 97\)](#).

You can enable **one** MFA device (of any kind) per root user or IAM user.

Note

If you want to enable the device from the command line, use `aws iam enable-mfa-device`. To enable the MFA device with the IAM API, use the `EnableMFADevice` action.

Enable a Hardware MFA Device for an IAM User (Console)

You can enable a hardware MFA device from the AWS Management Console.

To enable a hardware MFA device for an IAM user (console)

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, choose **Users**.
3. Choose the name of the user for whom you want to enable MFA, and then choose the **Security credentials** tab.
4. Next to **Assigned MFA device**, choose the pencil icon ().
5. In the **Manage MFA Device** wizard, choose **A hardware MFA device** and then choose **Next Step**.
6. Type the device serial number. The serial number is usually on the back of the device.
7. In the **Authentication Code 1** box, type the six-digit number displayed by the MFA device. You might need to press the button on the front of the device to display the number.



8. Wait 30 seconds while the device refreshes the code, and then type the next six-digit number into the **Authentication Code 2** box. You might need to press the button on the front of the device again to display the second number.
9. Choose **Next Step**.

Important

Submit your request immediately after generating the authentication codes. If you generate the codes and then wait too long to submit the request, the MFA device successfully associates with the user but the MFA device becomes out of sync. This happens because time-based one-time passwords (TOTP) expire after a short period of time. If this happens, you can [resync the device \(p. 102\)](#).

The device is ready for use with AWS. For information about using MFA with the AWS Management Console, see [Using MFA Devices With Your IAM Sign-in Page \(p. 56\)](#).

Enable a Hardware MFA Device for the AWS Account Root User (Console)

You can use IAM in the AWS Management Console to configure and enable a hardware MFA device for your root user. To manage MFA devices for the AWS account, you must be signed in to AWS using your AWS account root user credentials. You cannot manage MFA devices for the root user using other credentials.

If your MFA device is lost, stolen, or not working, you can still sign in using alternative factors of authentication. This means that if you can't sign in with your MFA device, you can sign in by verifying your identity using the email and phone that are registered with your account. Before you enable MFA for your root user, review your account settings and contact information to make sure that you have access to the email and phone number. To learn about signing in using alternative factors of authentication, see [What If an MFA Device Is Lost or Stops Working? \(p. 106\)](#). To disable this feature, contact [AWS Support](#).

Note

If you are using an AWS account created after September 14, 2017, you might see differences in the following console pages: **Sign in with authentication device** and **Troubleshoot your authentication device**. However, the same features are provided. In either case, if you cannot verify your account email address and phone number using alternative factors of authentication, contact [AWS Support](#) to deactivate your MFA setting.

To enable the MFA device for your root user (console)

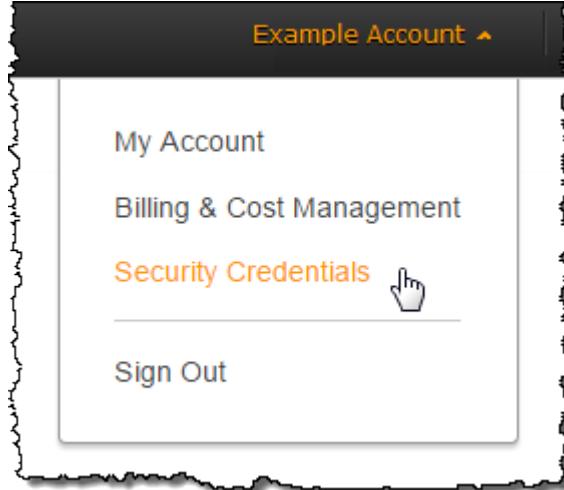
1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.

Important

To manage MFA devices for the AWS account, you must use your root user credentials to sign in to AWS. You cannot manage MFA devices for the root user while signed in with other credentials.

2. Do one of the following:

- **Option 1:** Choose **Dashboard**, and under **Security Status**, expand **Activate MFA on your root account**.
- **Option 2:** On the right side of the navigation bar, choose on your account name, and then choose **Security Credentials**. If necessary, choose **Continue to Security Credentials**. Then expand the **Multi-Factor Authentication (MFA)** section on the page.



3. Choose **Manage MFA** or **Activate MFA**, depending on which option you chose in the preceding step.
4. In the wizard, choose **A hardware MFA device** and then choose **Next Step**.
5. In the **Serial Number** box, type the serial number that is found on the back of the MFA device.
6. In the **Authentication Code 1** box, type the six-digit number displayed by the MFA device. You might need to press the button on the front of the device to display the number.



7. Wait 30 seconds while the device refreshes the code, and then type the next six-digit number into the **Authentication Code 2** box. You might need to press the button on the front of the device again to display the second number.
8. Choose **Next Step**. The MFA device is now associated with the AWS account.

Important

Submit your request immediately after generating the authentication codes. If you generate the codes and then wait too long to submit the request, the MFA device successfully associates with the user but the MFA device becomes out of sync. This happens because time-based one-time passwords (TOTP) expire after a short period of time. If this happens, you can [resync the device \(p. 102\)](#).

The next time you use your root user credentials to sign in, you must type a code from the MFA device.

Replace or "Rotate" a Physical MFA Device

You can have only one MFA device assigned to a user at a time. If the user loses a device or needs to replace it for any reason, you must first deactivate the old device. Then you can add the new device for the user.

- To deactivate the device currently associated with a user, see [Deactivating MFA Devices \(p. 104\)](#).
- To add a replacement hardware MFA device for an IAM user, follow the steps in the procedure [Enable a Hardware MFA Device for an IAM User \(Console\) \(p. 96\)](#) earlier in this topic.
- To add a replacement virtual MFA device for the AWS account root user, follow the steps in the procedure [Enable a Hardware MFA Device for the AWS Account Root User \(Console\) \(p. 97\)](#) earlier in this topic.

PREVIEW - Enabling SMS Text Message MFA Devices

AWS is no longer accepting new participants for the SMS MFA preview. We encourage you to use MFA on your AWS account through either a [hardware-based \(p. 96\)](#) or [virtual \(software-based\) \(p. 93\)](#) MFA token device.

If your account is already participating in the SMS MFA Preview program, you can continue using this feature.

An SMS MFA device can be any mobile device with a phone number that can receive standard [SMS text messages](#). When an MFA code is needed, AWS sends it to the phone number that is configured for the IAM user.

Note

SMS MFA can be used only with IAM users. It cannot be used with the AWS account root user. To protect the root user with MFA, you must use either a [hardware-based \(p. 96\)](#) or [virtual \(software-based\) \(p. 93\)](#) MFA token device.

Enable an SMS MFA Device for an IAM User (AWS Management Console)

Note

Currently, you can manage SMS MFA only in the AWS Management Console.

You can use IAM in the AWS Management Console to configure an IAM user with a phone number to enable SMS MFA.

To enable SMS MFA for an IAM user (console)

1. Use your AWS account ID or account alias, your IAM user name, and your password to sign in to the [IAM console](#).

Note

For your convenience, the AWS sign-in page uses a browser cookie to remember your IAM user name and account information. If you previously signed in as a different user, choose **Sign in to a different account** near the bottom of the page to return to the main sign-in page. From there, you can type your AWS account ID or account alias to be redirected to the IAM user sign-in page for your account.

2. In the navigation pane, choose **Users**.
3. In the **User Name** list, choose the name (not the check box) of the intended MFA user.
4. Choose the **Security credentials** tab. Next to **Assigned MFA device**, choose the pencil icon ().
5. In the **Manage MFA Device** wizard, choose **An SMS MFA device**, and then choose **Next Step**.

6. Type the phone number to which you want to send MFA codes for this IAM user, and then choose **Next Step**.
7. A six-digit authentication code is immediately sent to the specified phone number for verification. Type the six-digit code and then choose **Next Step**. If the code does not arrive in a reasonable amount of time), choose **Resend Code**. Note that SMS is not a service with a guaranteed delivery time.
8. If AWS successfully verifies the code, the wizard ends. Otherwise, choose **Finish** to close the wizard.

Change the Phone Number for SMS MFA for an IAM User

To change the phone number of the SMS MFA device assigned to an IAM user, you must delete the current MFA device. Then create a new device with the new phone number. To learn how to delete a device, see [Deactivating MFA Devices \(p. 104\)](#). To create a new device, see the previous procedures in this topic.

Enable and manage virtual MFA devices (AWS CLI, Tools for Windows PowerShell, or AWS API)

The following list shows the command line commands or API actions to use to enable a virtual MFA device.

Note

You must use the AWS Management Console to manage any MFA device for the AWS account root user in your AWS account. You cannot manage the MFA device for the root user with the AWS API, AWS CLI, Tools for Windows PowerShell, or any other command-line tool.

At this time you can manage SMS MFA devices only by using the AWS Management Console.

When you enable an MFA device from the AWS Management Console, the console performs many of these steps for you. If you instead create a virtual device using the AWS CLI, Tools for Windows PowerShell, or AWS API, then you must perform the steps manually and in the correct order. For example, to create a virtual MFA device, you must create the IAM object, extract the code as either a string or a QR code graphic, and then sync the device and associate it with an IAM user. See the **Examples** section of [New-IAMVirtualMFADevice](#) for more details. For a physical device, you skip the creation step and go directly to syncing the device and associating it with the user.

To create the virtual device entity in IAM to represent a virtual MFA device

These commands provide an ARN for the device that is used in place of a serial number in many of the following commands.

- AWS CLI: `aws iam create-virtual-mfa-device`
- Tools for Windows PowerShell: `New-IAMVirtualMFADevice`
- AWS API: `CreateVirtualMFADevice`

To enable an MFA device for use with AWS

These commands synchronize the device with AWS and associates it with a user or the root user. If the device is virtual, use the ARN of the virtual device as the serial number.

Important

Submit your request immediately after generating the authentication codes. If you generate the codes and then wait too long to submit the request, the MFA device successfully associates with the user but the MFA device becomes out of sync. This happens because time-based one-time passwords (TOTP) expire after a short period of time. If this happens, you can resynchronize the device using the commands described below.

- AWS CLI: `aws iam enable-mfa-device`

- Tools for Windows PowerShell: [Enable-IAMMFADevice](#)
- AWS API: [EnableMFADevice](#)

To deactivate a device

These commands disassociate the device from the user and deactivates it. If the device is virtual, use the ARN of the virtual device as the serial number. You must also separately delete the virtual device entity.

- AWS CLI: `aws iam deactivate-mfa-device`
- Tools for Windows PowerShell: [Disable-IAMMFADevice](#)
- AWS API: [DeactivateMFADevice](#)

To list virtual MFA device entities

- AWS CLI: `aws iam list-virtual-mfa-devices`
- Tools for Windows PowerShell: [Get-IAMVirtualMFADevice](#)
- AWS API: [ListVirtualMFADevices](#)

To resynchronize an MFA device

Use these commands if the devices is generating codes that are not accepted by AWS. If the device is virtual, use the ARN of the virtual device as the serial number.

- AWS CLI: `aws iam resync-mfa-device`
- Tools for Windows PowerShell: [Sync-IAMMFADevice](#)
- AWS API: [ResyncMFADevice](#)

To delete a virtual MFA device entity in IAM

After the device is disassociated from the user, you can delete the device entity.

- AWS CLI: `aws iam delete-virtual-mfa-device`
- Tools for Windows PowerShell: [Remove-IAMVirtualMFADevice](#)
- AWS API: [DeleteVirtualMFADevice](#)

Checking MFA Status

Use the IAM console to check whether an AWS account root user or IAM user has a valid MFA device enabled.

To check the MFA status of a root user

1. Sign in to the AWS Management Console with your root user credentials and then open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Check under **Security Status** to see whether MFA is enabled or disabled. If MFA has not been activated, an alert symbol () is displayed next to **Activate MFA on your root user**.

If you want to enable MFA for the account, see [Enable a Virtual MFA Device for Your AWS Account Root User \(Console\) \(p. 94\)](#) or [Enable a Hardware MFA Device for the AWS Account Root User \(Console\) \(p. 97\)](#).

To check the MFA status of IAM users

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Users**.
3. If necessary, add the **MFA** column to the users table by completing the following steps:
 - a. Above the table on the far right, choose the settings icon ().
 - b. In **Manage Columns**, select **MFA**.
 - c. (Optional) Clear the check box for any column headings that you do not want to appear in the users table.
 - d. Choose **Close** to return to the list of users.
4. The **MFA** column tells you about the MFA device that is enabled. If no MFA device is active for the user, the console displays **Not enabled**. If the user has an MFA device enabled, the **MFA** column shows the type of device that is enabled with a value of **Hardware**, **SMS**, or **Virtual**.
5. To view additional information about the MFA device for a user, choose the name of the user whose MFA status you want to check. Then choose the **Security credentials** tab.
6. If no MFA device is active for the user, the console displays **No** next to **Assigned MFA device**. If the user has an MFA device enabled, the **Assigned MFA device** item shows a value for the device:
 - The device serial number of a hardware device (usually the number from the back of the device), such as `GAHT12345678`
 - The ARN in AWS for an SMS device, such as `arn:aws:iam::123456789012:sms-mfa/username`
 - The ARN in AWS for a virtual device, such as `arn:aws:iam::123456789012:mfa/username`

If you want to change the current setting, choose the edit icon () next to **Assigned MFA Device**. For hardware device information, see [Enabling a Hardware MFA Device \(Console\) \(p. 96\)](#). For SMS device information, see [PREVIEW - Enabling SMS Text Message MFA Devices \(p. 99\)](#). For virtual device information, see [Enabling a Virtual Multi-factor Authentication \(MFA\) Device \(p. 93\)](#).

Resynchronize MFA Devices

As an AWS administrator, you can resynchronize your IAM user MFA devices if they get out of synchronization. If a user's device is not synchronized when they try to use it, the user's sign-in attempt fails and IAM prompts the user to resynchronize the device.

If your AWS account root user MFA device is not working, you can resynchronize your device using the IAM console with or without completing the sign-in process.

Resynchronize an MFA Device (IAM Console)

To resynchronize an MFA device for an IAM user (console)

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, choose **Users**, and then choose the name of the user whose MFA device needs to be resynchronized.
3. Choose the **Security credentials** tab. Next to **Assigned MFA device**, choose the pencil icon ().
4. In the **Manage MFA Device** wizard, choose **Resynchronize MFA device**, and then choose **Next Step**.
5. Type the next two sequentially generated codes from the device into **Authentication Code 1** and **Authentication Code 2**. Then choose **Next Step**.

Important

Submit your request immediately after generating the codes. If you generate the codes and then wait too long to submit the request, the request appears to work but the device remains out of sync. This happens because time-based one-time passwords (TOTP) expire after a short period of time.

To resynchronize your root user MFA before signing in (console)

1. On the **Amazon Web Services Sign In With Authentication Device** page, choose **Having problems with your authentication device? Click here.**

Note

If you are using an AWS account created after September 14, 2017, on the **Sign in with authentication device** page, choose **Troubleshoot your authentication device**.

2. In the **Re-Sync With Our Servers** section, type the next two sequentially generated codes from the device into **Authentication Code 1** and **Authentication Code 2**. Then choose **Re-sync authentication device**.
3. If required, type your password again and choose **Sign in**. Then complete the sign-in using your MFA device.

To resynchronize your root user MFA device after signing in (console)

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. On the right side of the navigation bar, choose your account name, and choose **Security Credentials**. If necessary, choose **Continue to Security Credentials**.
3. Expand the **Multi-Factor Authentication (MFA)** section on the page.
4. Next to your active MFA device, choose **Re-sync**.
5. In the **Manage MFA Device** dialog, type the next two sequentially generated codes from the device into **Authentication Code 1** and **Authentication Code 2**. Then choose **Next Step**.

Important

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

Resynchronize an MFA Device (AWS CLI)

To resynchronize an MFA device for an IAM user (AWS CLI)

- At a command prompt, issue the `aws iam resync-mfa-device` command:
 - Virtual MFA device: specify Amazon Resource Name (ARN) of device as `SerialNumber`.

```
$ aws iam resync-mfa-device --user-name Bob --serial-number
arn:aws:iam::123456789012:mfa/BobsMFA --authentication-code-1 123456 --
authentication-code-2 987654
```

- Physical MFA device: specify physical device's serial number as `SerialNumber`. The format is vendor specific.

```
PS C:\>Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Bob
```

Important

Submit your request immediately after generating the codes. If you generate the codes and then wait too long to submit the request, the request fails because the codes expire after a short time.

Tools for Windows PowerShell

To resynchronize an MFA device for an IAM user (Tools for Windows PowerShell)

- Use the [Sync-IAMMFADevice](#) cmdlet:
 - Virtual MFA device: specify Amazon Resource Name (ARN) of device as `SerialNumber`.

```
PS C:\>Sync-IAMMFADevice -UserName Bob -SerialNumber arn:aws:iam::123456789012:mfa/BobsMFA -AuthenticationCode1 123456 -AuthenticationCode2 987654
```

- Physical MFA device: specify physical device's serial number as `SerialNumber`. The format is vendor specific.

```
PS C:\>Sync-IAMMFADevice -UserName Bob -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -AuthenticationCode2 987654
```

Important

Submit your request immediately after generating the codes. If you generate the codes and then wait too long to submit the request, the request appears to work but the device remains out of sync. This happens because time-based one-time passwords (TOTP) expire after a short period of time.

IAM API

IAM has an API call that performs synchronization. In this case, we recommend that you give your MFA users permission to access this API call. You should build a tool based on that API call that lets your users resynchronize their devices whenever they need to.

To resynchronize an MFA device for an IAM user (API)

- Send the [ResyncMFADevice](#) request.

Deactivating MFA Devices

If an IAM user in your account is having trouble signing in with a multi-factor authentication (MFA) device, you can deactivate the device. This allows the user to sign in without using MFA. You might do this as a temporary solution while the MFA device is replaced, or if the device is temporarily unavailable. However, we recommend that you enable a new device for the user as soon as possible. To learn how to enable a new MFA device, see [the section called “Enabling MFA Devices” \(p. 92\)](#).

Note

If you use the API or CLI to delete a user from your AWS account, you must deactivate or delete the user's MFA device as part of the process of removing the user. For more information about deleting users, see [Managing IAM Users \(p. 68\)](#).

To deactivate an MFA device for a user (console)

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, choose **Users**.
3. To deactivate the MFA device for a user, choose the name of the user whose MFA you want to remove.
4. Choose the **Security credentials** tab. Next to **Assigned MFA device**, choose the pencil icon (✎).
5. In the **Manage MFA Device** wizard, choose **Deactivate MFA device**, and then choose **Next Step**.

The device is removed from AWS and cannot be used to sign in or authenticate requests until it is reactivated and associated with an AWS user or AWS account root user.

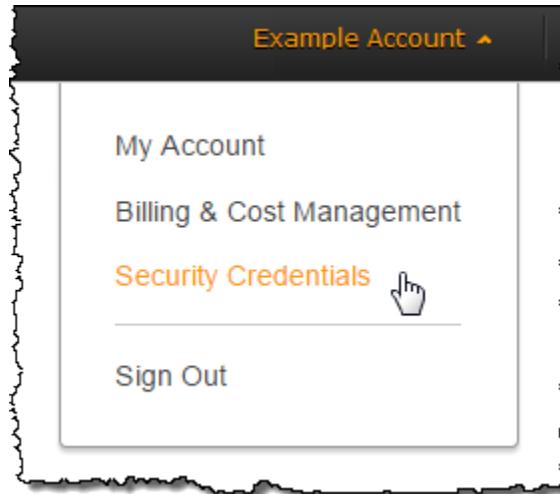
To deactivate the MFA device for your AWS account root user (console)

1. Use your AWS account root user credentials to sign in to the [AWS Management Console](#).

Important

To manage MFA devices for the AWS account, you must sign in to AWS with your AWS account root user credentials. You cannot manage MFA devices for the root user with other credentials.

2. On the navigation bar, choose your account name, and then choose **Security Credentials**. If a prompt appears, choose **Continue to Security Credentials**.



3. Expand the **Multi-Factor Authentication (MFA)** section.
4. In the row for the MFA device that you want to deactivate, choose **Deactivate**.

The MFA device is deactivated for the AWS account.

To deactivate an MFA device for a user (AWS CLI, Tools for Windows PowerShell, or AWS API)

- AWS CLI: `aws iam deactivate-mfa-device`
- Tools for Windows PowerShell: `Disable-IAMMFADevice`
- AWS API: `DeactivateMFADevice`

What If an MFA Device Is Lost or Stops Working?

If your AWS account root user multi-factor authentication (MFA) device is lost, damaged, or not working, you can sign in using alternative methods of authentication. This means that if you can't sign in with your MFA device, you can sign in by verifying your identity using the email and phone that are registered with your account.

If the device appears to be functioning properly, but you cannot use it to access your AWS resources, then it might be out of synchronization with the AWS system. For information about synchronizing an MFA device, see [Resynchronize MFA Devices \(p. 102\)](#).

If the MFA device associated with an IAM user is lost or stops working, the user can't recover it. IAM users must contact an administrator to deactivate the device.

Before you sign in as a root user using alternative factors of authentication, make sure that you have access to the email and phone number that are associated with your account.

To sign in using alternative factors of authentication as an AWS account root user

1. On the [Amazon Web Services Sign In With Authentication Device](#) page, choose **Having problems with your authentication device? Click here.**

Note

If you are using an AWS account created after September 14, 2017, you might see the following console text: **Sign in with authentication device, Troubleshoot your authentication device.** However, the same features are provided. In either case, if you cannot verify your account email address and phone number using alternative factors of authentication, contact [AWS Support](#) to deactivate your MFA setting.

2. If required, type your password again and choose **Sign in**.
3. In the **Sign In Using Alternative Factors of Authentication** section, choose **Sign in using alternative factors**.
4. To authenticate your account by verifying the email address, choose **Send verification email**.
5. Check the email that is associated with your AWS account for a message from Amazon Web Services (no-reply-aws@amazon.com). Follow the directions in the email.

If you don't see the email in your account, check your spam folder, or return to your browser and choose **Resend the email**.

6. After you verify your email address, you can continue authenticating your account. To verify your phone number, choose **Call me now**.
7. Answer the call from AWS and, when prompted, enter the 6-digit number from the AWS website on your phone keypad.

If you don't receive a call from AWS, choose **Sign in** to sign in to the console again and start over. Or choose [AWS Support](#) to contact support for help.

8. After you verify your phone number, you can sign in to your account by choosing **Sign in to the console**.
9. If you are using a hardware MFA device, contact the third-party provider for help fixing or replacing the device. You can continue to sign in using alternative factors of authentication until you receive your new device. After you have the new physical MFA device, go to the [AWS Security Credentials](#) page and delete the old MFA hardware device entity before you create a new one.

If you are using a virtual MFA device, remove the account from your device. Then go to the [AWS Security Credentials](#) page and delete the old MFA virtual device entity before you create a new one.

10. If your MFA device is missing or stolen, also [change your AWS password \(p. 75\)](#) in case an attacker has stolen the authentication device and might also have your current password.

To get help for an MFA device as an IAM user

1. Contact the AWS administrator or other person who gave you the user name and password for the IAM user. The administrator must deactivate the MFA device as described in [Deactivating MFA Devices \(p. 104\)](#) so that you can sign in.
 2. If you are using a hardware MFA device, contact the third-party provider for help fixing or replacing the device. After you have the new physical MFA device, enable the device as described in [Enabling a Hardware MFA Device \(Console\) \(p. 96\)](#).
- If you are using a virtual MFA device, remove the account from your device. Then enable the virtual device as described in [Enabling a Virtual Multi-factor Authentication \(MFA\) Device \(p. 93\)](#).
3. If your MFA device is missing or stolen, also [change your password \(p. 83\)](#) in case an attacker has stolen the authentication device and might also have your current password.

Configuring MFA-Protected API Access

With IAM policies, you can specify which APIs a user is allowed to call. In some cases, you might want the additional security of requiring a user to be authenticated with AWS multi-factor authentication (MFA) before the user is allowed to perform particularly sensitive actions.

For example, you might have a policy that allows a user to perform the Amazon EC2 `RunInstances`, `DescribeInstances`, and `StopInstances` actions. But you might want to restrict a destructive action like `TerminateInstances` and ensure that users can perform that action only if they authenticate with an AWS MFA device.

Topics

- [Overview \(p. 107\)](#)
- [Scenario: MFA Protection for Cross-Account Delegation \(p. 109\)](#)
- [Scenario: MFA Protection for Access to APIs in the Current Account \(p. 111\)](#)
- [Scenario: MFA Protection for Resources That Have Resource-based Policies \(p. 111\)](#)

Overview

Adding MFA protection to APIs involves these tasks:

1. The administrator configures an AWS MFA device for each user who needs to make API requests that require MFA authentication. This process is described at [Enabling MFA Devices \(p. 92\)](#).
2. The administrator creates policies for the users that include a `Condition` element that checks whether the user authenticated with an AWS MFA device.
3. The user calls one of the STS APIs that support the MFA parameters `AssumeRole` or `GetSessionToken`, depending on the scenario for MFA protection, as explained later. As part of the call, the user includes the device identifier for the device that's associated with the user, as well as the time-based one-time password (TOTP) that the device generates. In either case, the user gets back temporary security credentials that the user can then use to make additional requests to AWS.

Note

MFA protection for a service's APIs is available only if the service supports temporary security credentials. For a list of these services, see [Using Temporary Security Credentials to Access AWS](#).

If authorization fails, AWS returns an "Access Denied" error message (as it does for any unauthorized access). With MFA-protected API policies in place, AWS denies access to the APIs specified in the policies if the user attempts to use an API without valid MFA authentication or if the timestamp of the request

for the API is outside of the allowed range specified in the policy. The user must be reauthenticated with MFA by requesting new temporary security credentials with an MFA code and device serial number.

IAM Policies with MFA Conditions

Policies with MFA conditions can be attached to the following:

- An IAM user or group
- A resource such as an Amazon S3 bucket, Amazon SQS queue, or Amazon SNS topic
- The trust policy of an IAM role that can be assumed by a user

You can use an MFA condition in a policy to check the following properties:

- Existence—To simply verify that the user did authenticate with MFA, check that the `aws:MultiFactorAuthPresent` key is `True` in a `Bool` condition. The key is only present when the user authenticates with short term credentials. Long term credentials, such as access keys, do not include this key.
- Duration—if you want to grant access only within a specified time after MFA authentication, use a numeric condition type to compare the `aws:MultiFactorAuthAge` key's age to a value (such as 3600 seconds). Note that the `aws:MultiFactorAuthAge` key is not present if MFA was not used.

The following example shows the trust policy of an IAM role that includes an MFA condition to test for the existence of MFA authentication. With this policy, users from the AWS account specified in the `Principal` element (replace `ACCOUNT-B-ID` with a valid AWS account ID) can assume the role that this policy is attached to, but only if the user is MFA authenticated.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {"Effect": "Allow",  
         "Principal": {"AWS": "ACCOUNT-B-ID"},  
         "Action": "sts:AssumeRole",  
         "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}  
    ]  
}
```

For more information on the condition types for MFA, see [Available Global Condition Keys \(p. 477\)](#), [Numeric Condition Operators \(p. 442\)](#), and [Condition Operator to Check Existence of Condition Keys \(p. 447\)](#).

Choosing Between GetSessionToken and AssumeRole

AWS STS provides two APIs that let users pass MFA information: `GetSessionToken` and `AssumeRole`. The API that the user calls to get temporary security credentials depends on which of the following scenarios applies.

Use `GetSessionToken` for these scenarios:

- Call APIs that access resources in the same AWS account as the IAM user who makes the request. Note that temporary credentials from a `GetSessionToken` request can access IAM and STS APIs *only* if you include MFA information in the request for credentials. Because temporary credentials returned by `GetSessionToken` include MFA information, you can check for MFA in individual API calls made by the credentials.
- Access to resources that are protected with resource-based policies that include an MFA condition.

Use `AssumeRole` for these scenarios:

- Call APIs that access resources in the same or a different AWS account. The API calls can include any IAM or STS API. Note that to protect access you enforce MFA at the time when the user assumes the role. The temporary credentials returned by `AssumeRole` do not include MFA information in the context, so you cannot check individual API calls for MFA. This is why you must use `GetSessionToken` to restrict access to resources protected by resource-based policies.

Details about how to implement these scenarios are provided later in this document.

Important Points About MFA-Protected API Access

It's important to understand the following aspects of MFA protection for APIs:

- MFA protection is available only with temporary security credentials, which must be obtained with `AssumeRole` or `GetSessionToken`.
- You cannot use MFA-protected API access with AWS account root user credentials.
- Federated users cannot be assigned an MFA device for use with AWS services, so they cannot access AWS resources controlled by MFA. (See next point.)
- Other AWS STS APIs that return temporary credentials do not support MFA. For `AssumeRoleWithWebIdentity` and `AssumeRoleWithSAML`, the user is authenticated by an external provider and AWS cannot determine whether that provider required MFA. For `GetFederationToken`, MFA is not necessarily associated with a specific user.
- Similarly, long-term credentials (IAM user access keys and root user access keys) cannot be used with MFA-protected API access because they don't expire.
- `AssumeRole` and `GetSessionToken` can also be called without MFA information. In that case, the caller gets back temporary security credentials, but the session information for those temporary credentials does not indicate that the user authenticated with MFA.
- To establish MFA protection for APIs, you add MFA conditions to policies. If a policy doesn't include the condition for MFAs, the policy does not enforce the use of MFA. For cross-account delegation, if the role's trust policy doesn't include an MFA condition then there is no MFA protection for the API calls that are made with the role's temporary security credentials.
- When you allow another AWS account to access resources in your account, *even when you require multi-factor authentication*, the security of your resources depends on the configuration of the trusted account—that is, the other account (not yours). Any identity in the trusted account that has permission to create virtual MFA devices can construct an MFA claim to satisfy that part of your role's trust policy. Before you allow another account's access to your AWS resources that require multi-factor authentication, you should ensure that the trusted account's owner follows security best practices and restricts access to sensitive APIs—such as MFA device-management APIs—to specific, trusted identities.
- If a policy includes an MFA condition, a request is denied if users have not been MFA authenticated, or if they provide an invalid MFA device identifier or invalid TOTP.

Scenario: MFA Protection for Cross-Account Delegation

In this scenario, you want to delegate access to IAM users in another account, but only if the users are authenticated with an AWS MFA device. (For more information about cross-account delegation, see [Roles Terms and Concepts \(p. 135\)](#).)

Imagine that you have account A (the trusting account that owns the resource to be accessed), with the IAM user Alice, who has administrator permission. She wants to grant access to user Bob in account B (the trusted account), but wants to make sure that Bob is authenticated with MFA before he assumes the role.

1. In the trusting account A, Alice creates an IAM role named `CrossAccountRole` and sets the principal in the role's trust policy to the account ID of account B. The trust policy grants permission to the AWS

STS AssumeRole action. Alice also adds an MFA condition to the trust policy, as in the following example.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {"Effect": "Allow",  
         "Principal": {"AWS": "ACCOUNT-B-ID"},  
         "Action": "sts:AssumeRole",  
         "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}}  
    ]  
}
```

2. Alice adds a permission policy to the role that specifies what the role is allowed to do. The permission policy for a role with MFA protection is no different than any other role-permission policy. The following example shows the policy that Alice adds to the role; it allows an assuming user to perform any DynamoDB action on the table Books in account A.

Note

The permission policy does not include an MFA condition. It is important to understand that the MFA authentication is used only to determine whether a user can assume the role. Once the user has assumed the role, no further MFA checks are made.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {"Effect": "Allow",  
         "Action": ["dynamodb:*"],  
         "Resource": ["arn:aws:dynamodb:region:ACCOUNT-A-ID:table/Books"]}  
    ]  
}
```

3. In trusted account B, the administrator makes sure that IAM user Bob is configured with an AWS MFA device and that he knows the ID of the device—that is, the serial number if it's a hardware MFA device, or the device's ARN if it's a virtual MFA device.
4. In account B, the administrator attaches the following policy to user Bob (or a group that he's a member of) that allows him to call the AssumeRole action. The resource is set to the ARN of the role that Alice created in step 1. Notice that this policy does not contain an MFA condition.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {"Effect": "Allow",  
         "Action": ["sts:AssumeRole"],  
         "Resource": ["arn:aws:iam::ACCOUNT-A-ID:role/CrossAccountRole"]}  
    ]  
}
```

5. In account B, Bob (or an application that Bob is running) calls AssumeRole. The API call includes the ARN of the role to assume (`arn:aws:iam::ACCOUNT-A-ID:role/CrossAccountRole`), the ID of the MFA device, and the current TOTP that Bob gets from his device.

When Bob calls AssumeRole, AWS determines whether he has valid credentials, including the requirement for MFA. If so, Bob successfully assumes the role and can perform any DynamoDB action on the table named Books in account A while using the role's temporary credentials.

For an example of a program that calls AssumeRole, see [Calling AssumeRole with MFA Authentication \(Python\) \(p. 116\)](#).

Scenario: MFA Protection for Access to APIs in the Current Account

In this scenario, you want to make sure that a user in your AWS account can access sensitive API actions only when the user is authenticated using an AWS MFA device.

Imagine that you have account A that contains a group of developers who need to work with EC2 instances. Ordinary developers can work with the instances, but they are not granted permissions for the `ec2:StopInstances` or `ec2:TerminateInstances` actions. You want to limit those "destructive" privileged actions to just a few trusted users, so you add MFA protection to the policy that allows these sensitive Amazon EC2 actions.

In this scenario, one of those trusted users is user Carol. User Alice is an administrator in account A.

1. Alice makes sure that Carol is configured with an AWS MFA device and that Carol knows the ID of the device—the serial number if it's a hardware MFA device, or the device's ARN if it's a virtual MFA device.
2. Alice creates a group named `EC2-Admins` and adds user Carol to the group.
3. Alice attaches the following policy to the `EC2-Admins` group. This policy grants users permission to call the Amazon EC2 `StopInstances` and `TerminateInstances` actions only if the user has authenticated using MFA.

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": [  
            "ec2:StopInstances",  
            "ec2:TerminateInstances"  
        ],  
        "Resource": ["*"],  
        "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}  
    }]  
}
```

4. If user Carol needs to stop or terminate an Amazon EC2 instance, she (or an application that she is running) calls `GetSessionToken` passing the ID of the MFA device and the current TOTP that Carol gets from her device.
5. User Carol (or an application that Carol is using) uses the temporary credentials provided by `GetSessionToken` to call the Amazon EC2 `StopInstances` or `TerminateInstances` action.

For an example of a program that calls `GetSessionToken`, see [Calling GetSessionToken with MFA Authentication \(Python and C#\) \(p. 115\)](#) later in this document.

Scenario: MFA Protection for Resources That Have Resource-based Policies

In this scenario, you are the owner of an S3 bucket, an SQS queue, or an SNS topic and you want to make sure that any user from any AWS account who accesses the resource is authenticated by an AWS MFA device.

This scenario illustrates a way to provide cross-account MFA protection without requiring users to assume a role first. If the user is authenticated by MFA and is able to get temporary security credentials from `GetSessionToken`, and if the user's account is trusted by the resource's policy, the user can access the resource.

Imagine that you are in account A and you create an S3 bucket. You want to grant access to this bucket to users who are in several different AWS accounts, but only if those users are authenticated with MFA.

In this scenario, user Alice is an administrator in account A. User Charlie is an IAM user in account C.

1. In account A, Alice creates a bucket named Account-A-bucket.
2. Alice adds the bucket policy to the bucket. The policy allows any user in account A, account B, or account C to perform the S3 PutObject and DeleteObject actions in the bucket. The policy includes an MFA condition.

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Principal": {"AWS": [  
            "ACCOUNT-A-ID",  
            "ACCOUNT-B-ID",  
            "ACCOUNT-C-ID"  
        ]},  
        "Action": [  
            "s3:PutObject",  
            "s3:DeleteObject"  
        ],  
        "Resource": ["arn:aws:s3:::ACCOUNT-A-BUCKET-NAME/*"],  
        "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}  
    }]  
}
```

Note

Amazon S3 offers an MFA Delete feature for *root* account access (only). You can enable Amazon S3 MFA Delete when you set the versioning state of the bucket. Amazon S3 MFA Delete cannot be applied to an IAM user, and is managed independently from MFA-protected API access. An IAM user with permission to delete a bucket cannot delete a bucket with Amazon S3 MFA Delete enabled. For more information on Amazon S3 MFA Delete, see [MFA Delete](#).

3. In account C, an administrator makes sure that user Charlie is configured with an AWS MFA device and that he knows the ID of the device—the serial number if it's a hardware MFA device, or the device's ARN if it's a virtual MFA device.
4. In account C, Charlie (or an application that he is running) calls `GetSessionToken`. The call includes the ID or ARN of the MFA device and the current TOTP that Charlie gets from his device.
5. Charlie (or an application that he is using) uses the temporary credentials returned by `GetSessionToken` to call the Amazon S3 `PutObject` action to upload a file to Account-A-bucket.

For an example of a program that calls `GetSessionToken`, see [Calling GetSessionToken with MFA Authentication \(Python and C#\) \(p. 115\)](#) later in this document.

Note

The temporary credentials that `AssumeRole` returns won't work in this case because although the user can provide MFA information to assume a role, the temporary credentials returned by `AssumeRole` don't include the MFA information that is required in order to meet the MFA condition in the policy.

Sample Policies with MFA Conditions

The following examples show additional ways that MFA conditions can be added to policies. To learn how to create an IAM policy using these example JSON policy documents, see the section called “[Create a Policy on the JSON Tab](#)” (p. 320).

Note

The following examples show policies attached directly to an IAM user or group in your own AWS account. To adapt the example to MFA-protect APIs across accounts, you use IAM roles instead and put the MFA condition check in the role trust policy, not in the role access policy. For more information, see [Scenario: MFA Protection for Cross-Account Delegation \(p. 109\)](#).

Topics

- [Example 1: Granting Access After Recent MFA Authentication \(GetSessionToken\) \(p. 113\)](#)
- [Example 2: Denying Access to Specific APIs Without Valid MFA Authentication \(GetSessionToken\) \(p. 113\)](#)
- [Example 3: Denying Access to Specific APIs Without Recent Valid MFA Authentication \(GetSessionToken\) \(p. 114\)](#)

Example 1: Granting Access After Recent MFA Authentication (GetSessionToken)

The following example shows a policy attached to a user or group that grants Amazon EC2 access only if the user was authenticated with MFA within the last hour (3600 seconds). Note that if a user with long-term credentials and this policy calls the Amazon EC2 API, then the call fails because the `MultiFactorAuthAge` key is never present in the request context for long-term credentials. You can either allow long-term credentials by changing the operator to `NumericLessThanIfExists`, or you can require that the user get short-term credentials validated with an MFA using the `sts:GetSessionToken` API first.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["ec2:*"],
      "Resource": "*",
      "Condition": {"NumericLessThan": {"aws:MultiFactorAuthAge": "3600"}}
    }
  ]
}
```

Example 2: Denying Access to Specific APIs Without Valid MFA Authentication (GetSessionToken)

The following example shows a policy attached to a user or group that grants access to the entire Amazon EC2 API, but denies access to `StopInstances` and `TerminateInstances` if the user is not authenticated with MFA. The policy requires two statements to achieve the intended effect. The first statement (containing `"Sid": "AllowAllActionsForEC2"`) allows all Amazon EC2 actions. The second statement (containing `"Sid": "DenyStopAndTerminateWhenMFAIsNotPresent"`) denies the `StopInstances` and `TerminateInstances` actions when the MFA authentication context is missing (meaning MFA was not used).

Note

The condition check for `MultiFactorAuthPresent` in the Deny statement should not be a `{"Bool": {"aws:MultiFactorAuthPresent": false}}` because that key is not present and cannot be evaluated when MFA is not used. So instead, use the `BoolIfExists` check to see if the key is present before checking the value.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAllActionsForEC2",
      "Effect": "Allow",
      "Action": "ec2:*",
      "Resource": "*"
    },
    {
      "Sid": "DenyStopAndTerminateWhenMFAIsNotPresent",
      "Effect": "Deny",
      "Action": [
        "ec2:StopInstances",
        "ec2:TerminateInstances"
      ],
      "Condition": {"BoolIfExists": {"aws:MultiFactorAuthPresent": false}}
    }
  ]
}
```

```

        "ec2:TerminateInstances"
    ],
    "Resource": "*",
    "Condition": {"BoolIfExists": {"aws:MultiFactorAuthPresent": false}}
}
]
}

```

Example 3: Denying Access to Specific APIs Without *Recent* Valid MFA Authentication (`GetSessionToken`)

The following example shows a policy attached to a user or group that grants access to the entire Amazon EC2 API, but denies access to `StopInstances` and `TerminateInstances` unless the user authenticated with MFA within the last hour. This example expands on the previous example and requires three statements to achieve the intended effect. The first two statements are the same as the previous example. The second statement still contains the condition that denies `StopInstances` and `TerminateInstances` if MFA isn't used at all (the MFA context is missing). The third statement in the following example (containing `"Sid": "DenyStopAndTerminateWhenMFAIsOlderThanOneHour"`) contains an additional condition that denies the `StopInstances` and `TerminateInstances` actions when MFA authentication is present but occurred more than one hour prior to the request. For example, an IAM user might sign-in to the AWS Management Console with MFA and then attempt to stop or terminate an EC2 instance two hours later. The following policy prevents this. To stop or terminate an EC2 instance in this scenario, the user must sign out, sign in again with MFA, and then stop or terminate the instance within one hour of signing in.

Note

The condition check for `MultiFactorAuthPresent` in the first Deny statement uses `"BoolIfExists"`, because that key is not present and cannot be evaluated when MFA is not used. If MFA is not used and the value does not exist, it returns true and the statement matches and denies access.

The condition `aws:MultiFactorAuthAge` is only present when MFA context is present in the request. So statement 2 covers the case when MFA is not present at all, and statement 3 covers the case when MFA is present and evaluates whether it occurred in the proper time window. Again, when the key is not present, the `...IfExists` causes the test to return true, the statement matches, and the user is denied access to those APIs.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllActionsForEC2",
            "Effect": "Allow",
            "Action": "ec2:*",
            "Resource": "*"
        },
        {
            "Sid": "DenyStopAndTerminateWhenMFAIsNotPresent",
            "Effect": "Deny",
            "Action": [
                "ec2:StopInstances",
                "ec2:TerminateInstances"
            ],
            "Resource": "*",
            "Condition": {"BoolIfExists": {"aws:MultiFactorAuthPresent": false}}
        },
        {
            "Sid": "DenyStopAndTerminateWhenMFAIsOlderThanOneHour",
            "Effect": "Deny",
            "Action": [
                "ec2:StopInstances",
                "ec2:TerminateInstances"
            ]
        }
    ]
}

```

```
        ],
        "Resource": "*",
        "Condition": {"NumericGreaterThanIfExists": {"aws:MultiFactorAuthAge": "3600"}}
    }
}
```

Sample Code: Requesting Credentials with Multi-factor Authentication

The following examples show how to call `GetSessionTokenRole` and `AssumeRole` and pass MFA authentication. The credentials returned are then used to list all S3 buckets in the account.

Calling `GetSessionToken` with MFA Authentication (Python and C#)

The following examples, written using the [AWS SDK for Python \(Boto\)](#) and [AWS SDK for .NET](#), show how to call `GetSessionToken` and pass MFA authentication information. The temporary security credentials returned by `GetSessionTokenRole` are then used to list all S3 buckets in the account.

The policy attached to the user who runs this code (or to a group that the user is in) is assumed to include an MFA check. The policy also needs to grant the user permission to request the Amazon S3 `ListBuckets` action.

Using Python

```
import boto
from boto.s3.connection import S3Connection
from boto.sts import STSConnection

# Prompt for MFA time-based one-time password (TOTP)
mfa_TOTP = raw_input("Enter the MFA code: ")

# The calls to AWS STS GetSessionToken must be signed with the access key ID and secret
# access key of an IAM user. The credentials can be in environment variables or in
# a configuration file and will be discovered automatically
# by the STSConnection() function. For more information, see the Python SDK
# documentation: http://boto.readthedocs.org/en/latest/boto_config_tut.html

sts_connection = STSConnection()

# Use the appropriate device ID (serial number for hardware device or ARN for virtual
# device).
# Replace ACCOUNT-NUMBER-WITHOUT-HYPHENS and MFA-DEVICE-ID with appropriate values.

tempCredentials = sts_connection.get_session_token(
    duration=3600,
    mfa_serial_number=&region-arn;iam::ACCOUNT-NUMBER-WITHOUT-HYPHENS:mfa/MFA-DEVICE-ID",
    mfa_token=mfa_TOTP
)

# Use the temporary credentials to list the contents of an S3 bucket
s3_connection = S3Connection(
    aws_access_key_id=tempCredentials.access_key,
    aws_secret_access_key=tempCredentials.secret_key,
    security_token=tempCredentials.session_token
)

# Replace BUCKET-NAME with an appropriate value.
bucket = s3_connection.get_bucket(bucket_name="BUCKET-NAME")
objectlist = bucket.list()
for obj in objectlist:
```

```
    print obj.name
```

Using C#

```
Console.WriteLine("Enter MFA code: ");
string mfaTOTP = Console.ReadLine(); // Get string from user

/* The calls to AWS STS GetSessionToken must be signed using the access key ID and secret
   access key of an IAM user. The credentials can be in environment variables or in
   a configuration file and will be discovered automatically
   by the AmazonSecurityTokenServiceClient constructor. For more information, see
   http://docs.aws.amazon.com/AWSSdkDocsNET/latest/DeveloperGuide/net-dg-config-creds.html
*/
AmazonSecurityTokenServiceClient stsClient =
    new AmazonSecurityTokenServiceClient();
GetSessionTokenRequest getSessionTokenRequest = new GetSessionTokenRequest();
getSessionTokenRequest.DurationSeconds = 3600;

// Replace ACCOUNT-NUMBER-WITHOUT-HYPHENS and MFA-DEVICE-ID with appropriate values
getSessionTokenRequest.SerialNumber = "arn:aws:iam::ACCOUNT-NUMBER-WITHOUT-HYPHENS:mfa/MFA-
DEVICE-ID";
getSessionTokenRequest.TokenCode = mfaTOTP;

GetSessionTokenResponse getSessionTokenResponse =
    stsClient.GetSessionToken(getSessionTokenRequest);

// Extract temporary credentials from result of GetSessionToken call
GetSessionTokenResult getSessionTokenResult =
    getSessionTokenResponse.GetSessionTokenResult;
string tempAccessKeyId = getSessionTokenResult.Credentials.AccessKeyId;
string tempSessionToken = getSessionTokenResult.Credentials.SessionToken;
string tempSecretAccessKey = getSessionTokenResult.Credentials.SecretAccessKey;
SessionAWSCredentials tempCredentials = new SessionAWSCredentials(tempAccessKeyId,
    tempSecretAccessKey, tempSessionToken);

// Use the temporary credentials to list the contents of an S3 bucket
// Replace BUCKET-NAME with an appropriate value
ListObjectsRequest S3ListObjectsRequest = new ListObjectsRequest();
S3ListObjectsRequest.BucketName = "BUCKET-NAME";
S3Client = AWSClientFactory.CreateAmazonS3Client(tempCredentials);
ListObjectsResponse S3ListObjectsResponse =
    S3Client.ListObjects(S3ListObjectsRequest);
foreach (S3Object s3Object in S3ListObjectsResponse.S3Objects)
{
    Console.WriteLine(s3Object.Key);
}
```

Calling AssumeRole with MFA Authentication (Python)

The following example, written using the [AWS SDK for Python \(Boto\)](#), shows how to call `AssumeRole` and pass MFA authentication information. The temporary security credentials returned by `AssumeRole` are then used to list all Amazon S3 buckets in the account.

For more information about this scenario, see [Scenario: MFA Protection for Cross-Account Delegation \(p. 109\)](#).

```
import boto
from boto.s3.connection import S3Connection
from boto.sts import STSConnection

# Prompt for MFA time-based one-time password (TOTP)
mfa_TOTP = raw_input("Enter the MFA code: ")
```

```
# The calls to AWS STS AssumeRole must be signed with the access key ID and secret
# access key of an IAM user. (The AssumeRole API can also be called using temporary
# credentials, but this example does not show that scenario.)
# The IAM user credentials can be in environment variables or in
# a configuration file and will be discovered automatically
# by the STSConnection() function. For more information, see the Python SDK
# documentation: http://boto.readthedocs.org/en/latest/boto_config_tut.html

sts_connection = STSConnection()

# Use appropriate device ID (serial number for hardware device or ARN for virtual device)
# Replace ACCOUNT-NUMBER-WITHOUT-HYPHENS, ROLE-NAME, and MFA-DEVICE-ID with appropriate
# values
tempCredentials = sts_connectionassume_role(
    role_arn="arn:aws:iam::ACCOUNT-NUMBER-WITHOUT-HYPHENS:role/ROLE-NAME",
    role_session_name="AssumeRoleSession1",
    mfa_serial_number="arn:aws:iam::ACCOUNT-NUMBER-WITHOUT-HYPHENS:mfa/MFA-DEVICE-ID",
    mfa_token=mfa_TOTP
)

# Use the temporary credentials to list the contents of an S3 bucket
s3_connection = S3Connection(
    aws_access_key_id=tempCredentials.credentials.access_key,
    aws_secret_access_key=tempCredentials.credentials.secret_key,
    security_token=tempCredentials.credentials.session_token
)

# Replace BUCKET-NAME with a real bucket name
bucket = s3_connection.get_bucket(bucket_name="BUCKET-NAME")
objectlist = bucket.list()
for obj in objectlist:
    print obj.name
```

Finding Unused Credentials

To increase the security of your AWS account, remove IAM user credentials (that is, passwords and access keys) that are not needed. For example, when users leave your organization or no longer need AWS access, find the credentials that they were using and ensure that they are no longer operational. Ideally, you delete credentials if they are no longer needed. You can always recreate them at a later date if the need arises. At the very least you should change the password or deactivate the access keys so that the former users no longer have access.

Of course, the definition of *unused* can vary and usually means a credential that has not been used within a specified period of time.

Finding Unused Passwords

You can use the AWS Management Console to view password usage information for your users. If you have a large number of users, you can use the console to download a credential report with information about when each user last used their console password. You can also access the information from the AWS CLI, Tools for Windows PowerShell, or the IAM API.

To find unused passwords (console)

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, choose **Users**.
3. If necessary, add the **Console last sign-in** column to the users table:
 - a. Above the table on the far right, choose the settings icon ().

- b. In **Manage Columns**, select **Console last sign-in**.
- c. Choose **Close** to return to the list of users.
4. The **Console last sign-in** column shows the number of days since the user last signed in to AWS through the console. You can use this information to find users with passwords who have not signed in for more than a specified period of time. The column displays **Never** for users with passwords that have never signed in. **None** indicates users with no passwords. Passwords that have not been used recently might be good candidates for removal.

To find unused passwords by downloading the credentials report (console)

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, choose **Credential report**.
3. Choose **Download Report** to download a comma-separated value (CSV) file named `status_reports_<date>T<time>.csv`. The fifth column contains the `password_last_used` column with the dates or one of the following:
 - **N/A** – Users that do not have a password assigned at all.
 - **no_information** – Users that have not used their password since IAM began tracking password age on October 20, 2014.

To find unused passwords (API, CLI, PowerShell)

You can use the following commands to find unused passwords:

- **AWS CLI** – `aws iam list-users` returns a list of users, each with a `PasswordLastUsed` value. If the value is missing, then the user either has no password or the password has not been used since IAM began tracking password age on October 20, 2014.
- **Tools for Windows PowerShell** – `Get-IAMUsers` returns a collection of `User` objects, each of which has a `PasswordLastUsed` property. If the property value is `1/1/0001 12:00:00 AM`, then the user either has no password or the password has not been used since IAM began tracking password age on October 20, 2014.
- **IAM API** – `ListUsers` returns a collection of users, each of which has a `<PasswordLastUsed>` value. If the value is missing, then the user either has no password or the password has not been used since IAM began tracking password age on October 20, 2014.

For information about the commands to download the credentials report, see [Getting Credential Reports \(AWS CLI, Tools for Windows PowerShell, or IAM API\) \(p. 123\)](#).

Finding Unused Access Keys

You can use the AWS Management Console to view access key usage information for your users. If you have a large number of users, you can use the console to download a credentials report to find when each user last used their access keys. You can also access the information from the AWS CLI, Tools for Windows PowerShell, or the IAM API.

To find unused access keys (console)

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, choose **Users**.
3. If necessary, add the **Access key last used** column to the users table:
 - a. Above the table on the far right, choose the settings icon ().
 - b. In **Manage Columns**, select **Access key last used**.
 - c. Choose **Close** to return to the list of users.
4. The **Access key last used** column shows the number of days since the user last accessed AWS programmatically. You can use this information to find users with access keys that have not been used for more than a specified period of time. The column displays **None** for users with no access keys. Access keys that have not been used recently might be good candidates for removal.

To find unused access keys by downloading the credentials report (console)

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, choose **Credential Report**.
3. Choose **Download Report** to download a comma-separated value (CSV) file named `status_reports_<date>T<time>.csv`. Columns 11 through 13 contain the last used date, region, and service information for access key 1. Columns 16 through 18 contain the same information for access key 2. The value is **N/A** if the user does not have an access key or the user has not used the access key since IAM began tracking access key age on April 22, 2015.

To find unused access keys (API, CLI, PowerShell)

You can use the following commands to find unused access keys:

AWS CLI

- `aws iam list-access-keys` returns information about the access keys for a user, including the `AccessKeyId`.
- `aws iam get-access-key-last-used` takes an access key ID and returns output that includes the `LastUsedDate`, the `Region` in which the access key was last used, and the `ServiceName` of the last service requested. If the `LastUsedDate` field is missing, then the access key has not been used since IAM began tracking access key age on April 22, 2015.

Tools for Windows PowerShell

- `Get-IAMAccessKey` returns a collection of access key objects associated with the specified user. Each object has an `AccessKeyId` property.
- `Get-IAMAccessKeyLastUsed` takes an access key ID and returns an object with an `AccessKeyLastUsed` property object. The methods of that object include the `LastUsedDate`, the `Region` in which the access key was last used, and the `ServiceName` of the last service requested. If the property value is `1/1/0001 12:00:00 AM`, then the access key has not been used since IAM began tracking access key age on April 22, 2015.

IAM API

- `ListAccessKeys` returns a list of `AccessKeyId` values for access keys that are associated with the specified user.
- `GetAccessKeyLastUsed` takes an access key ID and returns a collection of values. Included are the `LastUsedDate`, the `Region` in which the access key was last used, and the `ServiceName` of the last service requested. If the value is missing, then user either has no access key or the access key has not been used since IAM began tracking access key age on April 22, 2015.

For information about the commands to download the credentials report, see [Getting Credential Reports \(AWS CLI, Tools for Windows PowerShell, or IAM API\) \(p. 123\)](#)

Getting Credential Reports for Your AWS Account

You can generate and download a *credential report* that lists all users in your account and the status of their various credentials, including passwords, access keys, and MFA devices. You can get a credential report from the AWS Management Console, the [AWS SDKs](#) and [Command Line Tools](#), or the IAM API.

You can use credential reports to assist in your auditing and compliance efforts. You can use the report to audit the effects of credential lifecycle requirements, such as password and access key rotation. You can provide the report to an external auditor, or grant permissions to an auditor so that he or she can download the report directly.

You can generate a credential report as often as once every four hours. When you request a report, IAM first checks whether a report for the AWS account has been generated within the past four hours. If so, the most recent report is downloaded. If the most recent report for the account is more than four hours old, or if there are no previous reports for the account, IAM generates and downloads a new report.

Required Permissions

- To create a credential report: `GenerateCredentialReport`
- To download the report: `GetCredentialReport`

Understanding the Report Format

Credential reports are formatted as comma-separated values (CSV) files. You can open CSV files with common spreadsheet software to perform analysis, or you can build an application that consumes the CSV files programmatically and performs custom analysis.

The CSV file contains the following columns:

user

The friendly name of the user.

arn

The Amazon Resource Name (ARN) of the user. For more information about ARNs, see [IAM ARNs \(p. 410\)](#).

user_creation_time

The date and time when the user was created, in [ISO 8601 date-time format](#).

password_enabled

When the user has a password, this value is `TRUE`. Otherwise it is `FALSE`. The value for the AWS account root user is always `not_supported`.

password_last_used

The date and time when the AWS account root user or IAM user's password was last used to sign in to an AWS website, in [ISO 8601 date-time format](#). AWS websites that capture a user's last sign-in time are the AWS Management Console, the AWS Discussion Forums, and the AWS Marketplace. When a password is used more than once in a 5-minute span, only the first use is recorded in this field.

- The value in this field is `no_information` in these cases:
 - The user's password has never been used.

- There is no sign-in data associated with the password, such as when user's password has not been used after IAM started tracking this information on October 20, 2014.
- The value in this field is **N/A** (not applicable) when the user does not have a password.

password_last_changed

The date and time when the user's password was last set, in [ISO 8601 date-time format](#). If the user does not have a password, the value in this field is **N/A** (not applicable). The value for the AWS account (root) is always **not_supported**.

password_next_rotation

When the account has a [password policy](#) that requires password rotation, this field contains the date and time, in [ISO 8601 date-time format](#), when the user is required to set a new password. The value for the AWS account (root) is always **not_supported**.

mfa_active

When a [multi-factor authentication \(p. 91\)](#) (MFA) device has been enabled for the user, this value is **TRUE**. Otherwise it is **FALSE**.

access_key_1_active

When the user has an access key and the access key's status is **Active**, this value is **TRUE**. Otherwise it is **FALSE**.

access_key_1_last_rotated

The date and time, in [ISO 8601 date-time format](#), when the user's access key was created or last changed. If the user does not have an active access key, the value in this field is **N/A** (not applicable).

access_key_1_last_used_date

The date and time, in [ISO 8601 date-time format](#), when the user's access key was most recently used to sign an AWS API request. When an access key is used more than once in a 15-minute span, only the first use is recorded in this field.

The value in this field is **N/A** (not applicable) in these cases:

- The user does not have an access key.
- The access key has never been used.
- The access key has not been used after IAM started tracking this information on April 22, 2015.

access_key_1_last_used_region

The [AWS region](#) in which the access key was most recently used. When an access key is used more than once in a 15-minute span, only the first use is recorded in this field.

The value in this field is **N/A** (not applicable) in these cases:

- The user does not have an access key.
- The access key has never been used.
- The access key was last used before IAM started tracking this information on April 22, 2015.
- The last used service is not region-specific, such as Amazon Simple Storage Service (Amazon S3).

access_key_1_last_used_service

The AWS service that was most recently accessed with the access key. The value in this field uses the service's [namespace](#)—for example, `s3` for Amazon S3 and `ec2` for Amazon Elastic Compute Cloud (Amazon EC2). When an access key is used more than once in a 15-minute span, only the first use is recorded in this field.

The value in this field is **N/A** (not applicable) in these cases:

- The user does not have an access key.

- The access key has never been used.
- The access key was last used before IAM started tracking this information on April 22, 2015.

access_key_2_active

When the user has a second access key and the second key's status is **Active**, this value is **TRUE**. Otherwise it is **FALSE**.

Note

Users can have up to two access keys, to make rotation easier. For more information about rotating access keys, see [Rotating Access Keys \(p. 88\)](#).

access_key_2_last_rotated

The date and time, in [ISO 8601 date-time format](#), when the user's second access key was created or last changed. If the user does not have a second active access key, the value in this field is **N/A** (not applicable).

access_key_2_last_used_date

The date and time, in [ISO 8601 date-time format](#), when the user's second access key was most recently used to sign an AWS API request. When an access key is used more than once in a 15-minute span, only the first use is recorded in this field.

The value in this field is **N/A** (not applicable) in these cases:

- The user does not have a second access key.
- The user's second access key has never been used.
- The user's second access key was last used before IAM started tracking this information on April 22, 2015.

access_key_2_last_used_region

The [AWS region](#) in which the user's second access key was most recently used. When an access key is used more than once in a 15-minute span, only the first use is recorded in this field. The value in this field is **N/A** (not applicable) in these cases:

- The user does not have a second access key.
- The user's second access key has never been used.
- The user's second access key was last used before IAM started tracking this information on April 22, 2015.
- The last used service is not region-specific, such as Amazon S3.

access_key_2_last_used_service

The AWS service that was most recently accessed with the user's second access key. The value in this field uses the service's [namespace](#)—for example, `s3` for Amazon S3 and `ec2` for Amazon Elastic Compute Cloud (Amazon EC2). When an access key is used more than once in a 15-minute span, only the first use is recorded in this field. The value in this field is **N/A** (not applicable) in these cases:

- The user does not have a second access key.
- The user's second access key has never been used.
- The user's second access key was last used before IAM started tracking this information on April 22, 2015.

cert_1_active

When the user has an X.509 signing certificate and that certificate's status is **Active**, this value is **TRUE**. Otherwise it is **FALSE**.

cert_1_last_rotated

The date and time, in [ISO 8601 date-time format](#), when the user's signing certificate was created or last changed. If the user does not have an active signing certificate, the value in this field is **N/A** (not applicable).

cert_2_active

When the user has a second X.509 signing certificate and that certificate's status is `Active`, this value is `TRUE`. Otherwise it is `FALSE`.

Note

Users can have up to two X.509 signing certificates, to make certificate rotation easier.

cert_2_last_rotated

The date and time, in [ISO 8601 date-time format](#), when the user's second signing certificate was created or last changed. If the user does not have a second active signing certificate, the value in this field is `N/A` (not applicable).

Getting Credential Reports (AWS Management Console)

You can use the AWS Management Console to download a credential report as a comma-separated values (CSV) file.

To download a credential report using the AWS Management Console

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, click **Credential report**.
3. Click **Download Report**.

Getting Credential Reports (AWS CLI, Tools for Windows PowerShell, or IAM API)

To generate a credential report

You can use the following commands to create a credential report:

- AWS CLI: `aws iam generate-credential-report`
- Tools for Windows PowerShell: `Request-IAMCredentialReport`
- IAM API: `GenerateCredentialReport`

To retrieve a credential report

You can use the following commands to retrieve a generated credential report:

- AWS CLI: `aws iam get-credential-report`
- Tools for Windows PowerShell: `Get-IAMCredentialReport`
- IAM API: `GetCredentialReport`

Using IAM with AWS CodeCommit: Git Credentials, SSH Keys, and AWS Access Keys

AWS CodeCommit is a managed version control service that hosts private Git repositories in the AWS cloud. To use AWS CodeCommit, you configure your Git client to communicate with AWS CodeCommit repositories. As part of this configuration, you provide IAM credentials that AWS CodeCommit can use to authenticate you. IAM supports AWS CodeCommit with three types of credentials:

- Git credentials, an IAM -generated user name and password pair you can use to communicate with AWS CodeCommit repositories over HTTPS.
- SSH keys, a locally generated public-private key pair that you can associate with your IAM user to communicate with AWS CodeCommit repositories over SSH.
- [AWS access keys \(p. 85\)](#), which you can use with the credential helper included with the AWS CLI to communicate with AWS CodeCommit repositories over HTTPS.

See the following sections for more information about each option.

Use Git Credentials and HTTPS with AWS CodeCommit (Recommended)

With Git credentials, you generate a static user name and password pair for your IAM user, and then use those credentials for HTTPS connections. You can also use these credentials with any third-party tool or integrated development environment (IDE) that supports static Git credentials.

Because these credentials are universal for all supported operating systems and compatible with most credential management systems, development environments, and other software development tools, this is the recommended method. You can reset the password for Git credentials at any time. You can also make the credentials inactive or delete them if you no longer need them.

Note

You cannot choose your own user name or password for Git credentials. IAM generates these credentials for you to help ensure they meet the security standards for AWS and secure repositories in AWS CodeCommit. You can download the credentials only once, at the time they are generated. Make sure that you save the credentials in a secure location. If necessary, you can reset the password at any time, but doing so invalidates any connections configured with the old password. You must reconfigure connections to use the new password before you can connect.

See the following topics for more information:

- To create an IAM user, see [Creating an IAM User in Your AWS Account \(p. 63\)](#).
- To generate and use Git credentials with AWS CodeCommit, see [For HTTPS Users Using Git Credentials in the AWS CodeCommit User Guide](#).

Note

Changing the name of an IAM user after generating Git credentials does not change the user name of the Git credentials. The user name and password remain the same and are still valid.

To rotate service specific credentials

1. Create a second service-specific credential set in addition to the set currently in use.
2. Update all of your applications to use the new set of credentials and validate that the applications are working.
3. Change the state of the original credentials to "Inactive".
4. Ensure that all of your applications are still working.
5. Delete the inactive service-specific credentials.

Use SSH Keys and SSH with AWS CodeCommit

With SSH connections, you create public and private key files on your local machine that Git and AWS CodeCommit use for SSH authentication. You associate the public key with your IAM user and store the private key on your local machine. See the following topics for more information:

- To create an IAM user, see [Creating an IAM User in Your AWS Account \(p. 63\)](#).
- To create an SSH public key and associate it with an IAM user, see [For SSH Connections on Linux, macOS, or Unix](#) or see [For SSH Connections on Windows](#) in the *AWS CodeCommit User Guide*.

Note

IAM accepts public keys in the OpenSSH RSA format only with a minimum length of 2048 bits and a maximum length of 16384 bits. If you provide your public key in another format, you will see an error message stating that the key format is not valid.

Use HTTPS with the AWS CLI Credential Helper and AWS CodeCommit

As an alternative to HTTPS connections with Git credentials, you can allow Git to use a cryptographically signed version of your IAM user credentials or Amazon EC2 instance role whenever Git needs to authenticate with AWS to interact with AWS CodeCommit repositories. This is the only connection method for AWS CodeCommit repositories that does not require an IAM user. This is also the only method that works with federated access and temporary credentials. Unless your business needs require federated access or the use of temporary credentials, creating and using IAM users for access is strongly recommended. See the following topics for more information:

- To learn more about federated access, see [Identity Providers and Federation \(p. 143\)](#) and [Providing Access to Externally Authenticated Users \(Identity Federation\) \(p. 141\)](#).
- To learn more about temporary credentials, see [Temporary Security Credentials \(p. 231\)](#) and [Temporary Access to AWS CodeCommit Repositories](#).

The AWS CLI credential helper is not compatible with other credential helper systems, such as Keychain Access or Windows Credential Management. There are additional configuration considerations when you configure HTTPS connections with the credential helper. For more information, see [For HTTPS Connections on Linux, macOS, or Unix with the AWS CLI Credential Helper](#) or [HTTPS Connections on Windows with the AWS CLI Credential Helper](#) in the *AWS CodeCommit User Guide*.

Working with Server Certificates

To enable HTTPS connections to your website or application in AWS, you need an *SSL/TLS server certificate*. You can use a server certificate provided by [AWS Certificate Manager \(ACM\)](#) or one that you obtained from an external provider. You can use ACM or IAM to store and deploy server certificates.

ACM is the preferred tool to provision, manage, and deploy your server certificates. With ACM you can request a certificate or deploy an existing ACM or external certificate to AWS resources. Certificates provided by ACM are free and automatically renew. You can use ACM to manage server certificates from the console or programmatically. For more information about using ACM, see the [AWS Certificate Manager User Guide](#).

Use IAM as a certificate manager only when you must support HTTPS connections in a region that is not supported by ACM. IAM securely encrypts your private keys and stores the encrypted version in IAM SSL certificate storage. IAM supports deploying server certificates in all regions, but you must obtain your certificate from an external provider for use with AWS. You cannot upload an ACM certificate to IAM. Additionally, you cannot manage your certificates from the IAM Console.

For more information about requesting an ACM certificate, see [Request a Certificate](#) in the *AWS Certificate Manager User Guide*.

For more information about importing third party certificates into ACM, see [Importing Certificates](#) in the *AWS Certificate Manager User Guide*.

For more information about uploading third party certificates to IAM, see the following topics.

Topics

- [Uploading a Server Certificate \(IAM API\) \(p. 126\)](#)
- [Retrieving a Server Certificate \(IAM API\) \(p. 127\)](#)
- [Listing Server Certificates \(IAM API\) \(p. 127\)](#)
- [Renaming a Server Certificate or Updating its Path \(IAM API\) \(p. 127\)](#)
- [Deleting a Server Certificate \(IAM API\) \(p. 127\)](#)
- [Troubleshooting \(p. 128\)](#)

Uploading a Server Certificate (IAM API)

To upload a server certificate to IAM, you must provide the certificate and its matching private key. When the certificate is not self-signed, you must also provide a certificate chain. (You don't need a certificate chain when uploading a self-signed certificate.) Before you upload a certificate, ensure that you have all these items and that they meet the following criteria:

- The certificate must be valid at the time of upload. You cannot upload a certificate before its validity period begins (the certificate's `NotBefore` date) or after it expires (the certificate's `NotAfter` date).
- The private key must be unencrypted. You cannot upload a private key that is protected by a password or passphrase. For help decrypting an encrypted private key, see [Troubleshooting \(p. 128\)](#).
- The certificate, private key, and certificate chain must all be PEM-encoded. For help converting these items to PEM format, see [Troubleshooting \(p. 128\)](#).

To use the [IAM API](#) to upload a certificate, send an `UploadServerCertificate` request. The following example shows how to do this with the [AWS Command Line Interface \(AWS CLI\)](#). The example assumes the following:

- The PEM-encoded certificate is stored in a file named `Certificate.pem`.
- The PEM-encoded certificate chain is stored in a file named `CertificateChain.pem`.
- The PEM-encoded, unencrypted private key is stored in a file named `PrivateKey.pem`.

To use the following example command, replace these file names with your own and replace `ExampleCertificate` with a name for your uploaded certificate. Type the command on one continuous line. The following example includes line breaks and extra spaces to make it easier to read.

```
$ aws iam upload-server-certificate --server-certificate-name ExampleCertificate  
--certificate-body file://Certificate.pem  
--certificate-chain file://CertificateChain.pem  
--private-key file://PrivateKey.pem
```

When the preceding command is successful, it returns metadata about the uploaded certificate, including its [Amazon Resource Name \(ARN\)](#), its friendly name, its identifier (ID), its expiration date, and more.

Note

If you are uploading a server certificate to use with Amazon CloudFront, you must specify a path using the `--path` option. The path must begin with `/cloudfront` and must include a trailing slash (for example, `/cloudfront/test/`).

To use the AWS Tools for Windows PowerShell to upload a certificate, use [Publish-IAMServerCertificate](#).

Retrieving a Server Certificate (IAM API)

To use the IAM API to retrieve a certificate, send a [GetServerCertificate](#) request. The following example shows how to do this with the AWS CLI. Replace *ExampleCertificate* with the name of the certificate to retrieve.

```
$ aws iam get-server-certificate --server-certificate-name ExampleCertificate
```

When the preceding command is successful, it returns the certificate, the certificate chain (if one was uploaded), and metadata about the certificate.

Note

You cannot download or retrieve a private key from IAM after you upload it.

To use the AWS Tools for Windows PowerShell to retrieve a certificate, use [Get-IAMServerCertificate](#).

Listing Server Certificates (IAM API)

To use the IAM API to list your uploaded server certificates, send a [ListServerCertificates](#) request. The following example shows how to do this with the AWS CLI.

```
$ aws iam list-server-certificates
```

When the preceding command is successful, it returns a list that contains metadata about each certificate.

To use the AWS Tools for Windows PowerShell to list your uploaded server certificates, use [Get-IAMServerCertificates](#).

Renaming a Server Certificate or Updating its Path (IAM API)

To use the IAM API to rename a server certificate or update its path, send an [UpdateServerCertificate](#) request. The following example shows how to do this with the AWS CLI.

To use the following example command, replace the old and new certificate names and the certificate path, and type the command on one continuous line. The following example includes line breaks and extra spaces to make it easier to read.

```
$ aws iam update-server-certificate --server-certificate-name ExampleCertificate  
--new-server-certificate-name CloudFrontCertificate  
--new-path /cloudfront/
```

When the preceding command is successful, it does not return any output.

To use the AWS Tools for Windows PowerShell to rename a server certificate or update its path, use [Update-IAMServerCertificate](#).

Deleting a Server Certificate (IAM API)

To use the IAM API to delete a server certificate, send a [DeleteServerCertificate](#) request. The following example shows how to do this with the AWS CLI.

To use the following example command, replace *ExampleCertificate* with the name of the certificate to delete.

```
$ aws iam delete-server-certificate --server-certificate-name ExampleCertificate
```

When the preceding command is successful, it does not return any output.

To use the AWS Tools for Windows PowerShell to delete a server certificate, use [Remove-IAMServerCertificate](#).

Troubleshooting

Before you can upload a certificate to IAM, you must make sure that the certificate, private key, and certificate chain are all PEM-encoded. You must also ensure that the private key is unencrypted. See the following examples.

Example PEM-encoded certificate

```
-----BEGIN CERTIFICATE-----  
Base64-encoded certificate  
-----END CERTIFICATE-----
```

Example PEM-encoded, unencrypted private key

```
-----BEGIN RSA PRIVATE KEY-----  
Base64-encoded private key  
-----END RSA PRIVATE KEY-----
```

Example PEM-encoded certificate chain

A certificate chain contains one or more certificates. The following example contains three certificates, but your certificate chain might contain more or fewer.

```
-----BEGIN CERTIFICATE-----  
Base64-encoded certificate  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
Base64-encoded certificate  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
Base64-encoded certificate  
-----END CERTIFICATE-----
```

If these items are not in the right format for uploading to IAM, you can use [OpenSSL](#) to convert them to the right format.

To convert a certificate or certificate chain from DER to PEM

Use the [OpenSSL x509 command](#), as in the following example. In the following example command, replace *Certificate.der* with the name of the file that contains your DER-encoded certificate. Replace *Certificate.pem* with the desired name of the output file to contain the PEM-encoded certificate.

```
$ openssl x509 -inform DER -in Certificate.der -outform PEM -out Certificate.pem
```

To convert a private key from DER to PEM

Use the [OpenSSL rsa command](#), as in the following example. In the following example command, replace *PrivateKey.der* with the name of the file that contains your DER-encoded private key. Replace *PrivateKey.pem* with the desired name of the output file to contain the PEM-encoded private key.

```
$ openssl rsa -inform DER -in PrivateKey.der -outform PEM -out PrivateKey.pem
```

To decrypt an encrypted private key (remove the password or passphrase)

Use the [OpenSSL rsa command](#), as in the following example. To use the following example command, replace *EncryptedPrivateKey.pem* with the name of the file that contains your encrypted private key. Replace *PrivateKey.pem* with the desired name of the output file to contain the PEM-encoded unencrypted private key.

```
$ openssl rsa -in EncryptedPrivateKey.pem -out PrivateKey.pem
```

To convert a certificate bundle from PKCS#12 (PFX) to PEM

Use the [OpenSSL pkcs12 command](#), as in the following example. In the following example command, replace *CertificateBundle.p12* with the name of the file that contains your PKCS#12-encoded certificate bundle. Replace *CertificateBundle.pem* with the desired name of the output file to contain the PEM-encoded certificate bundle.

```
$ openssl pkcs12 -in CertificateBundle.p12 -out CertificateBundle.pem -nodes
```

To convert a certificate bundle from PKCS#7 to PEM

Use the [OpenSSL pkcs7 command](#), as in the following example. In the following example command, replace *CertificateBundle.p7b* with the name of the file that contains your PKCS#7-encoded certificate bundle. Replace *CertificateBundle.pem* with the desired name of the output file to contain the PEM-encoded certificate bundle.

```
$ openssl pkcs7 -in CertificateBundle.p7b -print_certs -out CertificateBundle.pem
```

IAM Groups

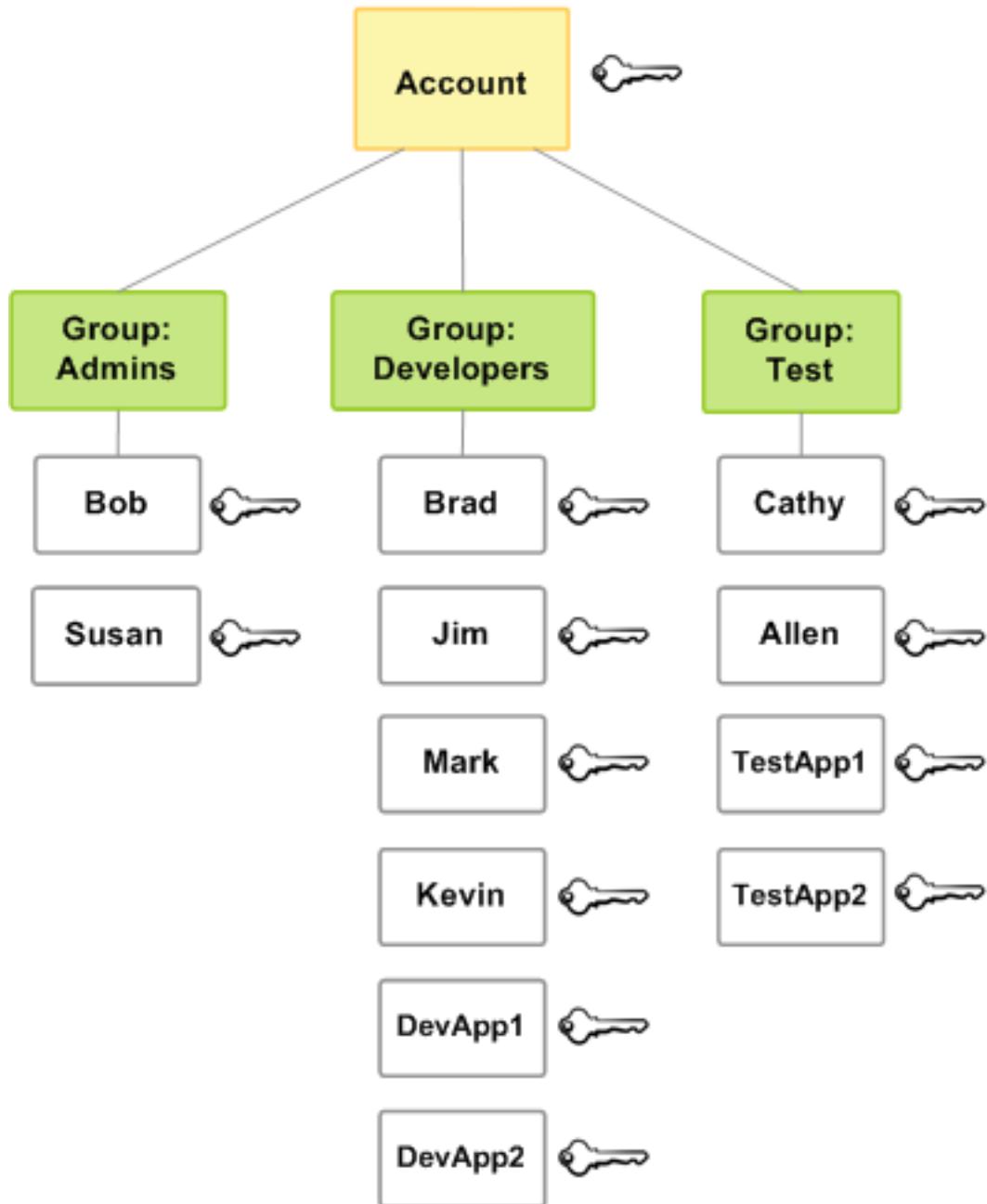
An IAM [group \(p. 129\)](#) is a collection of IAM users. Groups let you specify permissions for multiple users, which can make it easier to manage the permissions for those users. For example, you could have a group called *Admins* and give that group the types of permissions that administrators typically need. Any user in that group automatically has the permissions that are assigned to the group. If a new user joins your organization and needs administrator privileges, you can assign the appropriate permissions by adding the user to that group. Similarly, if a person changes jobs in your organization, instead of editing that user's permissions, you can remove him or her from the old groups and add him or her to the appropriate new groups.

Note that a group is not truly an "identity" in IAM because it cannot be identified as a `Principal` in a permission policy. It is simply a way to attach policies to multiple users at one time.

Following are some important characteristics of groups:

- A group can contain many users, and a user can belong to multiple groups.
- Groups can't be nested; they can contain only users, not other groups.
- There's no default group that automatically includes all users in the AWS account. If you want to have a group like that, you need to create it and assign each new user to it.
- There's a limit to the number of groups you can have, and a limit to how many groups a user can be in. For more information, see [Limitations on IAM Entities and Objects \(p. 414\)](#).

The following diagram shows a simple example of a small company. The company owner creates an **Admins** group for users to create and manage other users as the company grows. The **Admins** group creates a **Developers** group and a **Test** group. Each of these groups consists of users (humans and applications) that interact with AWS (Jim, Brad, DevApp1, and so on). Each user has an individual set of security credentials. In this example, each user belongs to a single group. However, users can belong to multiple groups.



Creating IAM Groups

To set up a group, you need to create the group, give it permissions based on the type of work that you expect the users in the group to do, and then add users to the group.

For information about the permissions that you need in order to create a group, see [Permissions Required to Access IAM Resources \(p. 372\)](#).

To create an IAM group and attach policies (AWS Management Console)

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, click **Groups** and then click **Create New Group**.
3. In the **Group Name** box, type the name of the group and then click **Next Step**.

Important

Group names must be unique within an account. They are not distinguished by case, for example, you cannot create groups named both "ADMINS" and "admins".

4. In the list of policies, select the check box for each policy that you want to apply to all members of the group. Then click **Next Step**.
5. Click **Create Group**.

For an example of how to set up an Administrators group, see [Creating Your First IAM Admin User and Group \(p. 17\)](#).

To create IAM groups and attach policies (AWS CLI, Tools for Windows PowerShell, AWS API)

Use one of the following commands to create a group:

- AWS CLI: `aws iam create-group`
- Tools for Windows PowerShell: `New-IAMGroup`
- AWS API: `CreateGroup`

Managing IAM Groups

Amazon Web Services offers multiple tools for managing IAM groups. For information about the permissions that you need in order to add and remove users in a group, see [Permissions Required to Access IAM Resources \(p. 372\)](#).

Topics

- [Listing IAM Groups \(p. 131\)](#)
- [Adding and Removing Users in an IAM Group \(p. 132\)](#)
- [Attaching a Policy to an IAM Group \(p. 132\)](#)
- [Renaming an IAM Group \(p. 133\)](#)
- [Deleting an IAM Group \(p. 133\)](#)

Listing IAM Groups

You can list all the groups in your account, list the users in a group, and list the groups a user belongs to. If you use the AWS CLI, Tools for Windows PowerShell, or AWS API, you can list all the groups with a particular path prefix

To list all the groups in your account

- [AWS Management Console](#): In the navigation pane, choose **Groups**.
- AWS CLI: `aws iam list-groups`
- Tools for Windows PowerShell: [Get-IAMGroups](#)
- AWS API: [ListGroups](#)

To list the users in a specific group

- [AWS Management Console](#): In the navigation pane, choose **Groups**, choose the name of the group, and then choose the **Users** tab.
- AWS CLI: `aws iam get-group`
- Tools for Windows PowerShell: [Get-IAMGroup](#)
- AWS API: [GetGroup](#)

To list all the groups that a user is in

- [AWS Management Console](#): In the navigation pane, chose **Users**, choose the user name, and then choose the **Groups** tab.
- AWS CLI: `aws iam list-groups-for-user`
- Tools for Windows PowerShell: [Get-IAMGroupForUser](#)
- AWS API: [ListGroupsForUser](#)

Adding and Removing Users in an IAM Group

At any time, you can add users to or remove users from an IAM group. This is useful as people enter and leave your organization.

To add a user to an IAM group

- [AWS Management Console](#): In the navigation pane, choose **Groups** and then choose the name of the group. Choose the **Users** tab and then choose **Add Users to Group**. Select the users you want to add and then choose **Add Users to Group**.
- AWS CLI: `aws iam add-user-to-group`
- Tools for Windows PowerShell: [Add-IAMUserToGroup](#)
- AWS API: [AddUserToGroup](#)

To remove a user from an IAM group

- [AWS Management Console](#): In the navigation pane, choose **Groups** and then choose the name of the group. Choose the **Users** tab and then choose **Remove Users from Group**. Select the users you want to add and then choose **Remove Users from Group**.
- AWS CLI: `aws iam remove-user-from-group`
- Tools for Windows PowerShell: [Remove-IAMUserFromGroup](#)
- AWS API: [RemoveUserFromGroup](#)

Attaching a Policy to an IAM Group

You can attach an [AWS managed policy \(p. 280\)](#)—that is, a prewritten policy provided by AWS—to a group, as explained in the following steps. To attach a customer managed policy—that is, a policy with

custom permissions that you create—you must first create the policy. For information about creating customer managed policies, see [Creating IAM Policies \(p. 317\)](#).

For more information about permissions and policies, see [Access Management \(p. 274\)](#).

To attach a policy to a group (AWS Management Console)

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, select **Policies**.
3. In the list of policies, select the check box next to the name of the policy to attach. You can use the **Filter** menu and the search box to filter the list of policies.
4. Click **Policy actions**, then click **Attach**.
5. For **Filter**, choose **All Types**, then click **Groups**.
6. Select the check box next to the name of the group to attach the policy to, then click **Attach policy**.

To attach a policy to a group (AWS CLI, Tools for Windows PowerShell, API)

- AWS CLI: `aws iam attach-group-policy`
- Tools for Windows PowerShell: [Register-IAMGroupPolicy](#)
- AWS API: [AttachGroupPolicy](#)

Renaming an IAM Group

When you change a group's name or path, the following happens:

- Any policies attached to the group stay with the group under the new name.
- The group retains all its users under the new name.
- The unique ID for the group remains the same. For more information about unique IDs, see [Unique IDs \(p. 413\)](#).

IAM does not automatically update policies that refer to the group as a resource to use the new name; you must manually do that. For example, let's say Bob is the manager of the testing part of the organization, and he has a policy attached to his IAM user entity that lets him add and remove users from the Test group. If an admin changes the name of the group to `Test_1` (or changes the path for the group), the admin also needs to update the policy attached to Bob to use the new name (or new path). Otherwise Bob won't be able to add and remove users from the group.

To change the name of an IAM group

- **AWS Management Console:** In the navigation pane, click **Groups** and then select the check box next to the group name. From the **Group Actions** list at the top of the page, select **Edit Group Name**. Type the new group name and then click **Yes, Edit**.
- AWS CLI: `aws iam update-group`
- Tools for Windows PowerShell: [Update-IAMGroup](#)
- AWS API: [UpdateGroup](#)

Deleting an IAM Group

When you delete a group in the AWS Management Console, the console automatically removes all group members, detaches all attached managed policies, and deletes all inline policies.

In contrast, when you use the AWS CLI, Tools for Windows PowerShell, or AWS API to delete a group, you must first remove the users in the group, delete any inline policies embedded in the group, and detach any managed policies attached to the group before you can delete the group.

To delete an IAM group (AWS Management Console)

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, select **Groups**.
3. In the list of groups, select the check box next to the name of the group to delete. You can use the **Filter** menu and the search box to filter the list of policies.
4. Click **Group Actions**, then click **Delete Group**.
5. In the confirmation box, click **Yes, Delete**.

To delete an IAM group (AWS CLI, Tools for Windows PowerShell, AWSAPI)

1. Remove all users from the group.
 - CLI: [aws iam get-group](#) (to get the list of users in the group), and [aws iam remove-user-from-group](#) (to remove a user from the group)
 - Tools for Windows PowerShell:

```
(Get-IAMGroup -GroupName "GroupToDelete").Users | Remove-IAMUserFromGroup -GroupName "GroupToDelete" -Force
```

- AWS API: [GetGroup](#) (to get the list of users in the group), and [RemoveUserFromGroup](#) (to remove a user from the group)
2. Delete all inline policies embedded in the group.
 - CLI: [aws iam list-group-policies](#) (to get a list of the group's inline policies), and [aws iam delete-group-policy](#) (to delete the group's inline policies)
 - Tools for Windows PowerShell:

```
Get-IAMGroupPolicies -GroupName "GroupToDelete" | % { Remove-IAMGroupPolicy -GroupName "GroupToDelete" -PolicyName $_ -Force }
```

- AWS API: [ListGroupPolicies](#) (to get a list of the group's inline policies), and [DeleteGroupPolicy](#) (to delete the group's inline policies)
3. Detach all managed policies attached to the group.
 - CLI: [aws iam list-attached-group-policies](#) (to get a list of the managed policies attached to the group), and [aws iam detach-group-policy](#) (to detach a managed policy from the group)
 - Tools for Windows PowerShell:

```
Get-IAMAttachedUserPolicies -UserName "UserToDelete" | % { Unregister-IAMUserPolicy -PolicyArn $_.PolicyArn -UserName "UserToDelete" -Force }
```

- AWS API: [ListAttachedGroupPolicies](#) (to get a list of the managed policies attached to the group), and [DetachGroupPolicy](#) (to detach a managed policy from the group)
4. Delete the group.
 - CLI: [aws iam delete-group](#)
 - Tools for Windows PowerShell: [Remove-IAMGroup](#)
 - AWS API: [DeleteGroup](#)

IAM Roles

An IAM *role* is similar to a user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials (password or access keys) associated with it. Instead, if a user assumes a role, [temporary security credentials \(p. 231\)](#) are created dynamically and provided to the user.

You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources. For example, you might want to grant users in your AWS account access to resources they don't usually have, or grant users in one AWS account access to resources in another account. Or you might want to allow a mobile app to use AWS resources, but not want to embed AWS keys within the app (where they can be difficult to rotate and where users can potentially extract them). Sometimes you want to give AWS access to users who already have identities defined outside of AWS, such as in your corporate directory. Or, you might want to grant access to your account to third parties so that they can perform an audit on your resources.

For these scenarios, you can delegate access to AWS resources using an *IAM role*. This section introduces roles and the different ways you can use them, when and how to choose among approaches, and how to create, manage, switch to (or assume), and delete roles.

Topics

- [Roles Terms and Concepts \(p. 135\)](#)
- [Common Scenarios for Roles: Users, Applications, and Services \(p. 137\)](#)
- [Identity Providers and Federation \(p. 143\)](#)
- [Using Service-Linked Roles \(p. 176\)](#)
- [Creating IAM Roles \(p. 184\)](#)
- [Using IAM Roles \(p. 205\)](#)
- [Managing IAM Roles \(p. 221\)](#)
- [How IAM Roles Differ from Resource-based Policies \(p. 229\)](#)

Roles Terms and Concepts

Here are some basic terms to help you get started with roles.

Role

A set of permissions that grant access to actions and resources in AWS. These permissions are attached to the role, not to an IAM user or group. Roles can be used by the following:

- An IAM user in the same AWS account as the role
- An IAM user in a different AWS account as the role
- A web service offered by AWS such as Amazon Elastic Compute Cloud (Amazon EC2)
- An external user authenticated by an external identity provider (IdP) service that is compatible with SAML 2.0 or OpenID Connect, or a custom-built identity broker.

AWS service role

A role that a service assumes to perform actions in your account on your behalf. When you set up most AWS service environments, you must define a role for the service to assume. This service role must include all the permissions required for the service to access the AWS resources that it needs.

Service roles vary from service to service, but many allow you to choose your permissions, as long as you meet the documented requirements for that service. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. You can create, modify, and delete a service role from within IAM.

AWS service role for an EC2 instance

A special type of service role that a service assumes to launch an Amazon EC2 instance that runs your application. This role is assigned to the EC2 instance when it is launched. AWS automatically provides temporary security credentials that are attached to the role and then makes them available for the EC2 instance to use on behalf of its applications. For details about using a service role for an EC2 instance, see [Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances \(p. 215\)](#).

AWS service-linked role

A unique type of service role that is linked directly to an AWS service. Service-linked roles are predefined by the service and include all the permissions that the service requires to call other AWS services on your behalf. The linked service also defines how you create, modify, and delete a service-linked role. A service might automatically create or delete the role. It might allow you to create, modify, or delete the role as part of a wizard or process in the service. Or it might require that you use IAM to create or delete the role. Regardless of the method, service-linked roles make setting up a service easier because you don't have to manually add the necessary permissions.

Note

If you are already using a service when it begins supporting service-linked roles, you might receive an email telling you about a new role in your account. In this case, the service automatically created the service-linked role in your account. You don't need to take any action to support this role, and you should not manually delete it. For more information, see [A New Role Appeared in My AWS Account \(p. 400\)](#).

For information about which services support using service-linked roles, see [AWS Services That Work with IAM \(p. 417\)](#) and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service. If the service does not include documentation for creating, modifying, or deleting the service-linked role, then you can use the IAM console, CLI, or API. For more information, see [Using Service-Linked Roles \(p. 176\)](#).

Delegation

The granting of permission to someone to allow access to resources that you control. Delegation involves setting up a trust between the account that owns the resource (the trusting account), and the account that contains the users that need to access the resource (the trusted account). The trusted and trusting accounts can be any of the following:

- The same account.
- Two accounts that are both under your (organization's) control.
- Two accounts owned by different organizations.

To delegate permission to access a resource, you [create an IAM role \(p. 184\)](#) that has two [policies \(p. 137\)](#) attached. The *permissions policy* grants the user of the role the needed permissions to carry out the intended tasks on the resource. The *trust policy* specifies which trusted accounts are allowed to grant its users permissions to assume the role.

When you create a trust policy, you cannot specify a wildcard (*) as a principal. The trust policy on the role in the trusting account is one-half of the permissions. The other half is a permissions policy attached to the user in the trusted account that [allows that user to switch to, or assume the role \(p. 205\)](#). A user who assumes a role temporarily gives up his or her own permissions and instead takes on the permissions of the role. When the user exits, or stops using the role, the original user permissions are restored. An additional parameter called [external ID \(p. 187\)](#) helps ensure secure use of roles between accounts that are not controlled by the same organization.

Federation

The creation of a trust relationship between an external identity provider and AWS. Users can sign in to a web identity provider, such as [Login with Amazon](#), [Facebook](#), [Google](#), or any IdP that is compatible with [OpenID Connect](#) (OIDC). Users can also sign in to an enterprise identity system that is compatible with Security Assertion Markup Language (SAML) 2.0, such as Microsoft Active Directory Federation Services. When you use OIDC and SAML 2.0 to configure a trust relationship between these external identity providers and AWS, the user is assigned to an IAM role and receives temporary credentials that enable the user to access your AWS resources.

Trust policy

A document in [JSON](#) format in which you define who is allowed to assume the role. This trusted entity is included in the policy as the *principal* element in the document. The document is written according to the rules of the [IAM policy language \(p. 425\)](#).

Permissions policy

A permissions document in [JSON](#) format in which you define what actions and resources the role can use. The document is written according to the rules of the [IAM policy language \(p. 425\)](#).

Principal

An entity in AWS that can perform actions and access resources. A principal can be an AWS account root user, an IAM user, or a role. You can grant permissions to access a resource in one of two ways:

- You can attach a permissions policy to a user (directly, or indirectly through a group) or to a role.
- For those services that support [resource-based policies \(p. 10\)](#), you can identify the principal in the `Principal` element of a policy attached to the resource.

If you reference an AWS account as principal, it generally means any principal defined within that account.

Note

You cannot use a wildcard (*) in the `Principal` element in a role's trust policy.

Role for cross-account access

Granting access to resources in one account to a trusted principal in a different account. Roles are the primary way to grant cross-account access. However, with some of the web services offered by AWS you can attach a policy directly to a resource (instead of using a role as a proxy). These are called resource-based policies, and you can use them to grant principals in another AWS account access to the resource. The following services support resource-based policies for the specified resources: Amazon Simple Storage Service (S3) buckets, Amazon Glacier vaults, Amazon Simple Notification Service (SNS) topics, and Amazon Simple Queue Service (SQS) queues. For more information, see [How IAM Roles Differ from Resource-based Policies \(p. 229\)](#).

Common Scenarios for Roles: Users, Applications, and Services

As with most AWS features, you generally have two ways to use a role: interactively in the IAM console, or programmatically with the AWS CLI, Tools for Windows PowerShell, or API.

- IAM users in your account using the IAM console can *switch to* a role to temporarily use the permissions of the role in the console. The users give up their original permissions and take on the permissions assigned to the role. When the users exit the role, their original permissions are restored.
- An application or a service offered by AWS (like Amazon EC2) can *assume* a role by requesting temporary security credentials for a role with which to make programmatic requests to AWS. You use a role this way so that you don't have to share or maintain long-term security credentials (for example, by creating an IAM user) for each entity that requires access to a resource.

Note

This guide uses the phrases *switch to a role* and *assume a role* interchangeably.

The simplest way to use roles is to grant your IAM users permissions to switch to roles that you create within your own or another AWS account. They can switch roles easily using the IAM console to use permissions that you don't ordinarily want them to have, and then exit the role to surrender those permissions. This can help prevent *accidental access* to or modification of sensitive resources.

For more complex uses of roles, such as granting access to applications and services, or federated external users, you can call the `AssumeRole` API. This API call returns a set of temporary credentials that the application can use in subsequent API calls. Actions attempted with the temporary credentials have only the permissions granted by the associated role. An application doesn't have to "exit" the role the way a user in the console does; rather the application simply stops using the temporary credentials and resumes making calls with the original credentials.

Federated users sign in by using credentials from an identity provider (IdP). AWS then provides temporary credentials to the trusted IdP to pass on to the user for including in subsequent AWS resource requests. Those credentials provide the permissions granted to the assigned role.

This section provides overviews of the following scenarios:

- [Provide access for an IAM user in one AWS account that you own to access resources in another account that you own \(p. 138\)](#)
- [Provide access to IAM users in AWS accounts owned by third parties \(p. 140\)](#)
- [Provide access for services offered by AWS to AWS resources \(p. 141\)](#)
- [Provide access for externally authenticated users \(identity federation\) \(p. 141\)](#)

Providing Access to an IAM User in Another AWS Account That You Own

You can grant your IAM users permission to switch to roles within your AWS account or to roles defined in other AWS accounts that you own.

Note

If you want to grant access to an account that you do not own or control, see [Providing Access to AWS Accounts Owned by Third Parties \(p. 140\)](#) later in this topic.

Imagine that you have Amazon Elastic Compute Cloud (Amazon EC2) instances that are critical to your organization. Instead of directly granting your users permission to terminate the instances, you can create a role with those privileges and allow administrators to switch to the role when they need to terminate an instance. This adds the following layers of protection to the instances:

- You must explicitly grant your users permission to assume the role.
- Your users must actively switch to the role using the AWS Management Console.
- You can add multi-factor authentication (MFA) protection to the role so that only users who sign in with an MFA device can assume the role.

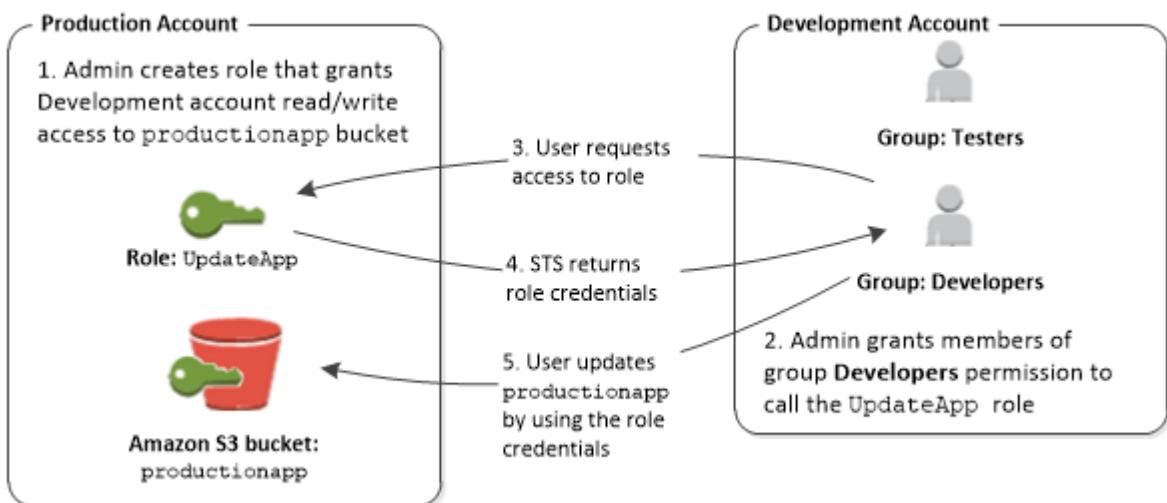
We recommend using this approach to enforce the *principle of least access*, that is, restricting the use of elevated permissions to only those times when they are needed for specific tasks. With roles you can help prevent accidental changes to sensitive environments, especially if you combine them with [auditing \(p. 262\)](#) to help ensure that roles are only used when needed.

When you create a role for this purpose, you specify the accounts by ID whose users need access in the `Principal` element of the role's trust policy. You can then grant specific users in those other accounts permissions to switch to the role.

A user in one account can switch to a role in the same or a different account. While using the role, the user can perform only the actions and access only the resources permitted by the role; their original user permissions are suspended. When the user exits the role, the original user permissions are restored.

Example Scenario Using Separate Development and Production Accounts

Imagine that your organization has multiple AWS accounts to isolate a development environment from a production environment. Users in the development account might occasionally need to access resources in the production account, such as when you are promoting an update from the development environment to the production environment. Although you could create separate identities (and passwords) for users who work in both accounts, managing credentials for multiple accounts makes identity management difficult. In the following figure, all users are managed in the development account, but some developers require limited access to the production account. The development account has two groups: Testers and Developers, and each group has its own policy.



1. In the production account an administrator uses IAM to create the `UpdateAPP` role in that account. In the role, the administrator defines a trust policy that specifies the development account as a `Principal`, meaning that authorized users from the development account can use the `UpdateAPP` role. The administrator also defines a permissions policy for the role that specifies that users of the role have read and write permissions to the Amazon Simple Storage Service (S3) bucket named `productionapp`.

The administrator then shares the account number and name of the role (for AWS console users) or the Amazon Resource Name (ARN) (for AWS CLI, Tools for Windows PowerShell, or AWS API access) of the role with anyone who needs to assume the role. The role ARN might look like `arn:aws:iam::123456789012:role/UpdateAPP`, where the role is named `UpdateAPP` and the role was created in account number `123456789012`.

Note

The administrator can optionally configure the role so that users who assume the role must first be authenticated using multi-factor authentication (MFA). For more information, see [Configuring MFA-Protected API Access \(p. 107\)](#).

2. In the development account an administrator grants members of the `Developers` group permission to switch to the role. This is done by granting the `Developers` group permission to call the AWS Security

Token Service (AWS STS) `AssumeRole` API for the `UpdateAPP` role. Any IAM user that belongs to the Developers group in the development account can now switch to the `UpdateAPP` role in the production account. Other users who are not in the developer group do not have permission to switch to the role and therefore cannot access the S3 bucket in the production account.

3. The user requests switches to the role:

- AWS console: The user chooses the account name on the navigation bar and chooses **Switch Role**. The user specifies the account ID (or alias) and role name. Alternatively, the user can click on a link sent in email by the administrator. The link takes the user to the **Switch Role** page with the details already filled in.
- AWS API/Tools for Windows PowerShell/AWS CLI: A user in the Developers group of the development account calls the `AssumeRole` function to obtain credentials for the `UpdateAPP` role. The user specifies the ARN of the `UpdateAPP` role as part of the call. If a user in the Testers group makes the same request, the request fails because Testers do not have permission to call `AssumeRole` for the `UpdateAPP` role ARN.

4. AWS STS returns temporary credentials:

- AWS console: AWS STS verifies the request with the role's trust policy to ensure that the request is from a trusted entity (which it is: the development account). After verification, AWS STS returns **temporary security credentials** to the AWS console.
- API/CLI: AWS STS verifies the request against the role's trust policy to ensure that the request is from a trusted entity (which it is: the Development account). After verification, AWS STS returns **temporary security credentials** to the application.

5. The temporary credentials allow access to the AWS resource:

- AWS console: The AWS console uses the temporary credentials on behalf of the user on all subsequent console actions, in this case, to read and write to the `productionapp` bucket. The console cannot access any other resource in the production account. When the user exits the role, the user's permissions revert to the original permissions held before switching to the role.
- API/CLI: The application uses the temporary security credentials to update the `productionapp` bucket. With the temporary security credentials, the application can only read from and write to the `productionapp` bucket and cannot access any other resource in the Production account. The application does not have to exit the role, but instead stops using the temporary credentials and uses the original credentials in subsequent API calls.

Providing Access to AWS Accounts Owned by Third Parties

When third parties require access to your organization's AWS resources, you can use roles to delegate access to them. For example, a third party might provide a service for managing your AWS resources. With IAM roles, you can grant these third parties access to your AWS resources without sharing your AWS security credentials. Instead, the third party can access your AWS resources by assuming a role that you create in your AWS account.

Third parties must provide you with the following information for you to create a role that they can assume:

- The third party's AWS account ID. You specify their AWS account ID as the principal when you define the trust policy for the role.
- An external ID to uniquely associate with the role. The external ID can be any secret identifier that is known by you and the third party. For example, you can use an invoice ID between you and the third party, but do not use something that can be guessed, like the name or phone number of the third party. You must specify this ID when you define the trust policy for the role. The third party must provide this ID when they assume the role. For more information about the external ID, see [How to Use an External ID When Granting Access to Your AWS Resources to a Third Party \(p. 187\)](#).
- The permissions that the third party requires to work with your AWS resources. You must specify these permissions when defining the role's permission policy. This policy defines what actions they can take and what resources they can access.

After you create the role, you must provide the role's Amazon Resource Name (ARN) to the third party. They require your role's ARN in order to assume the role.

For details about creating a role to delegate access to a third party, see [How to Use an External ID When Granting Access to Your AWS Resources to a Third Party \(p. 187\)](#).

Important

When you grant third parties access to your AWS resources, they can access any resource that you specify in the policy. Their use of your resources is billed to you. Ensure that you limit their use of your resources appropriately.

Providing Access to an AWS Service

Many AWS services require that you use roles to control what that service can access. A role that a service assumes to perform actions on your behalf is called a [service role \(p. 135\)](#). When a role serves a specialized purpose for a service, it can be categorized as a [service role for EC2 instances \(p. 136\)](#), or a [service-linked role \(p. 136\)](#). See the [AWS documentation](#) for each service to see if it uses roles and to learn how to assign a role for the service to use.

For details about creating a role to delegate access to a service offered by AWS, see [Creating a Role to Delegate Permissions to an AWS Service \(p. 191\)](#).

Providing Access to Externally Authenticated Users (Identity Federation)

Your users might already have identities outside of AWS, such as in your corporate directory. If those users need to work with AWS resources (or work with applications that access those resources), then those users also need AWS security credentials. You can use an IAM role to specify permissions for users whose identity is federated from your organization or a third-party identity provider (IdP).

Federating Users of a Mobile or Web-based App with Amazon Cognito

If you create a mobile or web-based app that accesses AWS resources, the app needs security credentials in order to make programmatic requests to AWS. For most mobile application scenarios, we recommend that you use [Amazon Cognito](#). You can use this service with the [AWS Mobile SDK for iOS](#) and the [AWS Mobile SDK for Android and Fire OS](#) to create unique identities for users and authenticate them for secure access to your AWS resources. Amazon Cognito supports the same identity providers as those listed in next section, and it also supports [developer authenticated identities](#) and unauthenticated (guest) access. Amazon Cognito also provides APIs for synchronizing user data so that it is preserved as users move between devices. For more information, see [Using Amazon Cognito for Mobile Apps \(p. 144\)](#).

Federating Users with Public Identity Service Providers or OpenID Connect

Whenever possible, use Amazon Cognito for mobile and web-based application scenarios. Amazon Cognito does most of the behind-the-scenes work with public identity provider services for you. It works with the same third-party services and also supports anonymous sign-ins. However, for more advanced scenarios, you can work directly with a third-party service like Login with Amazon, Facebook, Google, or any IdP that is compatible with OpenID Connect (OIDC). For more information about using web identity federation using one of these services, see [About Web Identity Federation \(p. 143\)](#).

Federating users with SAML 2.0

If your organization already uses an identity provider software package that supports SAML 2.0 (Security Assertion Markup Language 2.0), you can create trust between your organization as an identity provider

(IdP) and AWS as the service provider. You can then use SAML to provide your users with federated single-sign on (SSO) to the AWS Management Console or federated access to call AWS APIs. For example, if your company uses Microsoft Active Directory and Active Directory Federation Services, then you can federate using SAML 2.0. For more information about federating users with SAML 2.0, see [About SAML 2.0-based Federation \(p. 149\)](#).

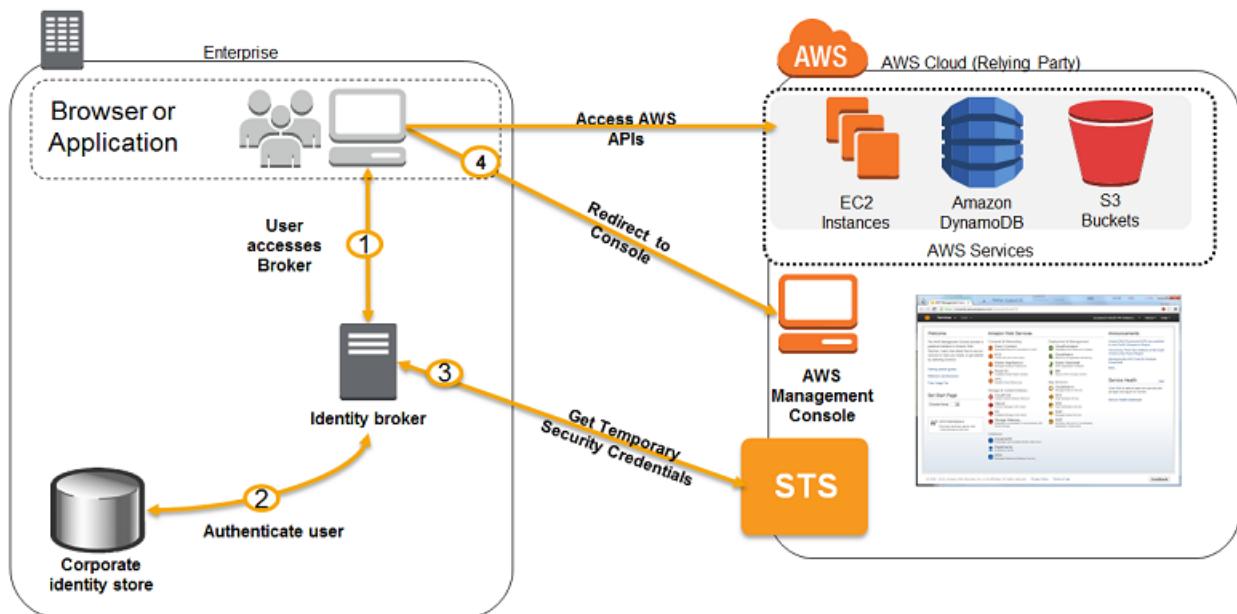
Federating users by creating a custom identity broker application

If your identity store is not compatible with SAML 2.0, then you can build a custom identity broker application to perform a similar function. The broker application authenticates users, requests temporary credentials for users from AWS, and then provides them to the user to access AWS resources.

For example, Example Corp. has many employees who need to run internal applications that access the company's AWS resources. The employees already have identities in the company identity and authentication system, and Example Corp. doesn't want to create a separate IAM user for each company employee.

Bob is a developer at Example Corp. To enable Example Corp. internal applications to access the company's AWS resources, Bob develops a custom identity broker application. The application verifies that employees are signed into the existing Example Corp. identity and authentication system, which might use LDAP, Active Directory, or another system. The identity broker application then obtains temporary security credentials for the employees. This scenario is similar to the previous one (a mobile app that uses a custom authentication system), except that the applications that need access to AWS resources all run within the corporate network, and the company has an existing authentication system.

To get temporary security credentials, the identity broker application calls either `AssumeRole` or `GetFederationToken` to obtain temporary security credentials, depending on how Bob wants to manage the policies for users and when the temporary credentials should expire. (For more information about the differences between these APIs, see [Temporary Security Credentials \(p. 231\)](#) and [Controlling Permissions for Temporary Security Credentials \(p. 246\)](#).) The call returns temporary security credentials consisting of an AWS access key ID, a secret access key, and a session token. The identity broker application makes these temporary security credentials available to the internal company application. The app can then use the temporary credentials to make calls to AWS directly. The app caches the credentials until they expire, and then requests a new set of temporary credentials. The following figure illustrates this scenario.



This scenario has the following attributes:

- The identity broker application has permissions to access IAM's token service (STS) API to create temporary security credentials.
- The identity broker application is able to verify that employees are authenticated within the existing authentication system.
- Users are able to get a temporary URL that gives them access to the AWS Management Console (which is referred to as single sign-on).

To see a sample application similar to the identity broker application that is described in this scenario, go to [Identity Federation Sample Application for an Active Directory Use Case](#) at *AWS Sample Code & Libraries*. For information about creating temporary security credentials, see [Requesting Temporary Security Credentials \(p. 233\)](#). For more information about federated users getting access to the AWS Management Console, see [Enabling SAML 2.0 Federated Users to Access the AWS Management Console \(p. 167\)](#).

Identity Providers and Federation

If you already manage user identities outside of AWS, you can use IAM *identity providers* instead of creating IAM users in your AWS account. With an identity provider (IdP), you can manage your user identities outside of AWS and give these external user identities permissions to use AWS resources in your account. This is useful if your organization already has its own identity system, such as a corporate user directory. It is also useful if you are creating a mobile app or web application that requires access to AWS resources.

When you use an IdP, you don't have to create custom sign-in code or manage your own user identities; the IdP provides that for you. Your external users sign in through a well-known identity provider, such as Login with Amazon, Facebook, Google, and many others. You can give those external identities permissions to use AWS resources in your account. Identity providers help keep your AWS account secure because you don't have to distribute or embed long-term security credentials, such as IAM access keys, in your application.

To use an IdP, you create an IAM identity provider entity to establish a trust relationship between your AWS account and the IdP. IAM supports IdPs that are compatible with [OpenID Connect \(OIDC\)](#) or [SAML 2.0 \(Security Assertion Markup Language 2.0\)](#). For more information about using one of these IdPs with AWS, see the following sections:

- [About Web Identity Federation \(p. 143\)](#)
- [About SAML 2.0-based Federation \(p. 149\)](#)

For details about creating the identity provider entity in IAM to establish a trust relationship between a compatible IdP and AWS, see [Creating IAM Identity Providers \(p. 153\)](#)

About Web Identity Federation

Imagine that you are creating a mobile app that accesses AWS resources, such as a game that runs on a mobile device and stores player and score information using Amazon S3 and DynamoDB.

When you write such an app, you'll make requests to AWS services that must be signed with an AWS access key. However, we **strongly recommend** that you do **not** embed or distribute long-term AWS credentials with apps that a user downloads to a device, even in an encrypted store. Instead, build your app so that it requests temporary AWS security credentials dynamically when needed using *web identity federation*. The supplied temporary credentials map to an AWS role that has only the permissions needed to perform the tasks required by the mobile app.

With web identity federation, you don't need to create custom sign-in code or manage your own user identities. Instead, users of your app can sign in using a well-known identity provider (IdP) —such as Login with Amazon, Facebook, Google, or any other [OpenID Connect \(OIDC\)](#)-compatible IdP, receive an authentication token, and then exchange that token for temporary security credentials in AWS that map to an IAM role with permissions to use the resources in your AWS account. Using an IdP helps you keep your AWS account secure, because you don't have to embed and distribute long-term security credentials with your application.

For most scenarios, we recommend that you use [Amazon Cognito](#) because it acts as an identity broker and does much of the federation work for you. For details, see the following section, [Using Amazon Cognito for Mobile Apps \(p. 144\)](#).

If you don't use Amazon Cognito, then you must write code that interacts with a web IdP (Login with Amazon, Facebook, Google, or any other OIDC-compatible IdP) and then calls the `AssumeRoleWithWebIdentity` API to trade the authentication token you get from those IdPs for AWS temporary security credentials. If you have already used this approach for existing apps, you can continue to use it.

Topics

- [Using Amazon Cognito for Mobile Apps \(p. 144\)](#)
- [Using Web Identity Federation APIs for Mobile Apps \(p. 146\)](#)
- [Identifying Users with Web Identity Federation \(p. 147\)](#)
- [Additional Resources for Web Identity Federation \(p. 149\)](#)

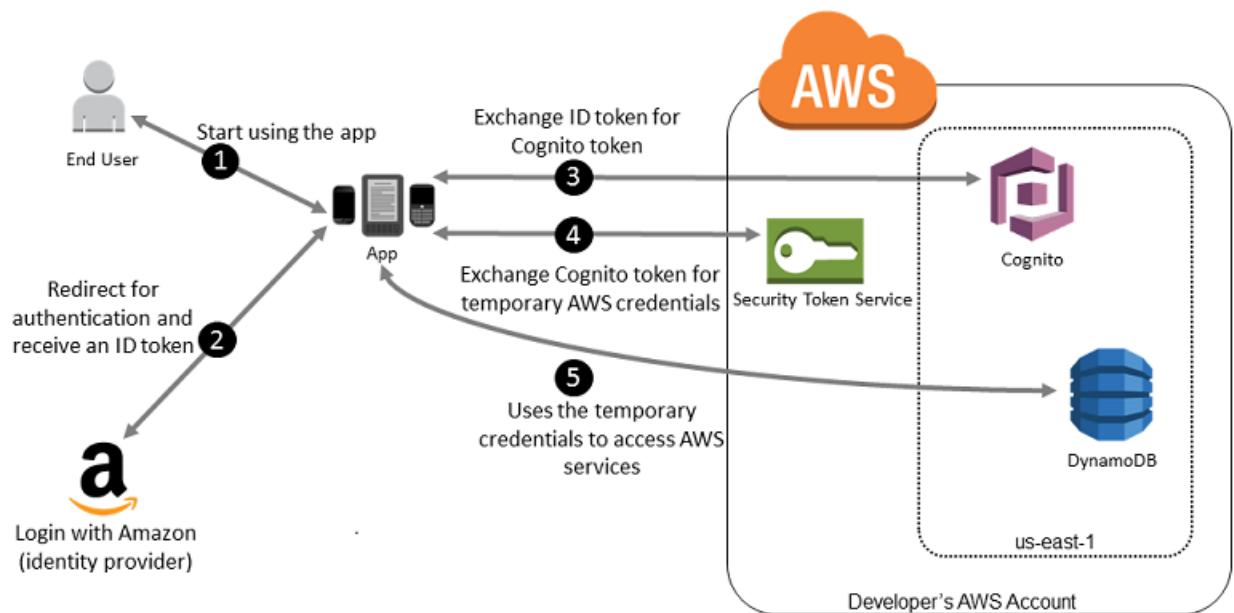
Using Amazon Cognito for Mobile Apps

The preferred way to use web identity federation is to use [Amazon Cognito](#). For example, Adele the developer is building a game for a mobile device where user data such as scores and profiles is stored in Amazon S3 and Amazon DynamoDB. Adele could also store this data locally on the device and use Amazon Cognito to keep it synchronized across devices. She knows that for security and maintenance reasons, long-term AWS security credentials should not be distributed with the game. She also knows that the game might have a large number of users. For all of these reasons, she does not want to create new user identities in IAM for each player. Instead, she builds the game so that users can sign in using an identity that they've already established with a well-known identity provider, such as [Login with Amazon](#), [Facebook](#), [Google](#), or any [OpenID Connect \(OIDC\)](#)-compatible identity provider. Her game can take advantage of the authentication mechanism from one of these providers to validate the user's identity.

To enable the mobile app to access her AWS resources, Adele first registers for a developer ID with her chosen IdPs. She also configures the application with each of these providers. In her AWS account that contains the Amazon S3 bucket and DynamoDB table for the game, Adele uses Amazon Cognito to create IAM roles that precisely define permissions that the game needs. If she is using an OIDC IdP, she also creates an IAM OIDC identity provider entity to establish trust between her AWS account and the IdP.

In the app's code, Adele calls the sign-in interface for the IdP that she configured previously. The IdP handles all the details of letting the user sign in, and the app gets an OAuth access token or OIDC ID token from the provider. Adele's app can trade this authentication information for a set of temporary security credentials that consist of an AWS access key ID, a secret access key, and a session token. The app can then use these credentials to access web services offered by AWS. The app is limited to the permissions that are defined in the role that it assumes.

The following figure shows a simplified flow for how this might work, using Login with Amazon as the IdP. For Step 2, the app can also use Facebook, Google, or any OIDC-compatible identity provider, but that's not shown here.



1. A customer starts your app on a mobile device. The app asks the user to sign in.
2. The app uses Login with Amazon resources to accept the user's credentials.
3. The app uses Cognito APIs to exchange the Login with Amazon ID token for a Cognito token.
4. The app requests temporary security credentials from AWS STS, passing the Cognito token.
5. The temporary security credentials can be used by the app to access any AWS resources required by the app to operate. The role associated with the temporary security credentials and its assigned policies determines what can be accessed.

Use the following process to configure your app to use Amazon Cognito to authenticate users and give your app access to AWS resources. For specific steps to accomplish this scenario, consult the documentation for Amazon Cognito.

1. (Optional) Sign up as a developer with Login with Amazon, Facebook, Google, or any other OpenID Connect (OIDC)-compatible identity provider and configure one or more apps with the provider. This step is optional because Amazon Cognito also supports unauthenticated (guest) access for your users.
2. Go to [Amazon Cognito in the AWS Management Console](#). Use the Amazon Cognito wizard to create an identity pool, which is a container that Amazon Cognito uses to keep end user identities organized for your apps. You can share identity pools between apps. When you set up an identity pool, Amazon Cognito creates one or two IAM roles (one for authenticated identities, and one for unauthenticated "guest" identities) that define permissions for Amazon Cognito users.
3. Download and integrate the [AWS SDK for iOS](#) or the [AWS SDK for Android](#) with your app, and import the files required to use Amazon Cognito.
4. Create an instance of the Amazon Cognito credentials provider, passing the identity pool ID, your AWS account number, and the Amazon Resource Name (ARN) of the roles that you associated with the identity pool. The Amazon Cognito wizard in the AWS Management Console provides sample code to help you get started.
5. When your app accesses an AWS resource, pass the credentials provider instance to the client object, which passes temporary security credentials to the client. The permissions for the credentials are based on the role or roles that you defined earlier.

For more information, see the following:

- [Amazon Cognito Identity](#) in the *AWS Mobile SDK for Android Developer Guide*.
- [Amazon Cognito Identity](#) in the *AWS Mobile SDK for iOS Developer Guide*.

Using Web Identity Federation APIs for Mobile Apps

For best results, use Amazon Cognito as your identity broker for almost all web identity federation scenarios. Amazon Cognito is easy to use and provides additional capabilities like anonymous (unauthenticated) access, and synchronizing user data across devices and providers. However, if you have already created an app that uses web identity federation by manually calling the `AssumeRoleWithWebIdentity` API, you can continue to use it and your apps will still work fine.

Note

To help understand how web identity federation works, you can use the [Web Identity Federation Playground](#). This interactive website lets you walk through the process of authenticating via Login with Amazon, Facebook, or Google, getting temporary security credentials, and then using those credentials to make a request to AWS.

The process for using web identity federation **without** Amazon Cognito follows this general outline:

1. Sign up as a developer with the IdP and configure your app with the IdP, who gives you a unique ID for your app. (Different IdPs use different terminology for this process. This outline uses the term *configure* for the process of identifying your app with the IdP.) Each IdP gives you an app ID that's unique to that IdP, so if you configure the same app with multiple IdPs, your app will have multiple app IDs. You can configure multiple apps with each provider.

The following external links provide information about using some of the commonly used identity providers:

- [Login with Amazon Developer Center](#)
- [Add Facebook Login to Your App or Website](#) on the Facebook developers site.
- [Using OAuth 2.0 for Login \(OpenID Connect\)](#) on the Google developers site.

Note

Although Amazon Cognito and Google are based on OIDC technology, you don't have to create an identity provider entity in IAM to use them. Support for Amazon Cognito and Google are built-in to AWS.

2. If you use an IdP that is compatible with OIDC, then create an identity provider entity for it in IAM.
3. In IAM, [create one or more roles](#) (p. 194). For each role, define who can assume the role (the trust policy) and what permissions the app's users are to have (the permissions policy). Typically, you create one role for each IdP that an app supports. For example, you might create a role that is assumed by an app when the user signs in through Login with Amazon, a second role for the same app where the user signs in through Facebook, and a third role for the app where the user signs in through Google. For the trust relationship, specify the IdP (like Amazon.com) as the `Principal` (the trusted entity), and include a Condition that matches the IdP assigned app ID. Examples of roles for different providers are described later in this topic.
4. In your application, authenticate your users with the IdP. The specifics of how to do this vary both according to which IdP you're using (Login with Amazon, Facebook, or Google) and on which platform your app. For example, an Android app's method of authentication can differ from that of an iOS app or a JavaScript-based web app.

Typically, if the user is not already signed in, the IdP takes care of displaying a sign-in page. After the IdP authenticates the user, the IdP returns an authentication token with information about the user to your app. The information included depends on what the IdP exposes and what information the user is willing to share. You can use this information in your app.

5. In your app, make an *unsigned* call to the `AssumeRoleWithWebIdentity` action to request temporary security credentials. In the request, you pass the IdP's authentication token and specify the Amazon Resource Name (ARN) for the IAM role that you created for that IdP. AWS verifies that the

token is trusted and valid and if so, returns temporary security credentials to your app that have the permissions for the role that you name in the request. The response also includes metadata about the user from the IdP, such as the unique user ID that the IdP associates with the user.

6. Using the temporary security credentials from the `AssumeRoleWithWebIdentity` response, your app makes signed requests to AWS APIs. The user ID information from the identity provider can distinguish users in your app—for example, you can put objects into Amazon S3 folders that include the user ID as prefixes or suffixes. This lets you create access control policies that lock the folder so only the user with that ID can access it. For more information, see [Identifying Users with Web Identity Federation \(p. 147\)](#) later in this topic.
7. Your app should cache the temporary security credentials so that you do not have to get new ones each time the app needs to make a request to AWS. By default, the credentials are good for one hour. When the credentials expire (or before then), you make another call to `AssumeRoleWithWebIdentity` to obtain a new set of temporary security credentials. Depending on the IdP and how they manage their tokens, you might have to refresh the IdP's token before you make a new call to `AssumeRoleWithWebIdentity`, since the IdP's tokens also usually expire after a fixed time. If you use the AWS SDK for iOS or the AWS SDK for Android, you can use the `AmazonSTSCredentialsProvider` action, which manages the IAM temporary credentials, including refreshing them as required.

Identifying Users with Web Identity Federation

When you create access policies in IAM, it's often useful to be able to specify permissions based on configured apps and on the ID of users who have authenticated using an identity provider. For example, your mobile app that's using web identity federation might keep information in Amazon S3 using a structure like this:

```
myBucket/app1/user1
myBucket/app1/user2
myBucket/app1/user3
...
myBucket/app2/user1
myBucket/app2/user2
myBucket/app2/user3
...
```

You might also want to additionally distinguish these paths by provider. In that case, the structure might look like the following (only two providers are listed to save space):

```
myBucket/Amazon/app1/user1
myBucket/Amazon/app1/user2
myBucket/Amazon/app1/user3
...
myBucket/Amazon/app2/user1
myBucket/Amazon/app2/user2
myBucket/Amazon/app2/user3

myBucket/Facebook/app1/user1
myBucket/Facebook/app1/user2
myBucket/Facebook/app1/user3
...
myBucket/Facebook/app2/user1
myBucket/Facebook/app2/user2
myBucket/Facebook/app2/user3
...
```

For these structures, `app1` and `app2` represent different apps, such as different games, and each user of the app has a distinct folder. The values for `app1` and `app2` might be friendly names that you assign (for

example, mynumbersgame) or they might be the app IDs that the providers assign when you configure your app. If you decide to include provider names in the path, those can also be friendly names like Cognito, Amazon, Facebook, and Google.

You can typically create the folders for app1 and app2 through the AWS Management Console, since the application names are static values. That's true also if you include the provider name in the path, since the provider name is also a static value. In contrast, the user-specific folders (`user1`, `user2`, `user3`, etc.) have to be created at run time from the app, using the user ID that's available in the `SubjectFromWebIdentityToken` value that is returned by the request to `AssumeRoleWithWebIdentity`.

To write policies that allow exclusive access to resources for individual users, you can match the complete folder name, including the app name and provider name, if you're using that. You can then include the following provider-specific context keys that reference the user ID that the provider returns:

- `cognito-identity.amazonaws.com:sub`
- `www.amazon.com:user_id`
- `graph.facebook.com:id`
- `accounts.google.com:sub`

For OIDC providers, use the fully qualified URL of the OIDC provider with the subcontext key, like the following example:

- `server.example.com:sub`

The following example shows a permission policy that grants access to a bucket in Amazon S3 only if the prefix for the bucket matches the string:

`myBucket/Amazon/mynumbersgame/user1`

The example assumes that the user is signed in using Login with Amazon, and that the user is using an app called mynumbersgame. The user's unique ID is presented as an attribute called `user_id`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["s3>ListBucket"],
            "Resource": ["arn:aws:s3:::myBucket"],
            "Condition": {"StringLike": {"s3:prefix": ["Amazon/mynumbersgame/${www.amazon.com:user_id}/*"]}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3>DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket/Amazon/mynumbersgame/${www.amazon.com:user_id}",
                "arn:aws:s3:::myBucket/Amazon/mynumbersgame/${www.amazon.com:user_id}/*"
            ]
        }
    ]
}
```

You would create similar policies for users who sign in using Amazon Cognito, Facebook, Google, or another OpenID Connect-compatible IdP. Those policies would use a different provider name as part of the path as well as different app IDs.

For more information about the web identity federation keys available for condition checks in policies, see [Available Keys for Web Identity Federation \(p. 480\)](#).

Additional Resources for Web Identity Federation

The following resources can help you learn more about web identity federation:

- [Amazon Cognito Identity](#) in the *AWS Mobile SDK for Android Developer Guide* and [Amazon Cognito Identity](#) in the *AWS Mobile SDK for iOS Developer Guide*.
- The [Web Identity Federation Playground](#) is an interactive website that lets you walk through the process of authenticating via Login with Amazon, Facebook, or Google, getting temporary security credentials, and then using those credentials to make a request to AWS.
- The entry [Web Identity Federation using the AWS SDK for .NET](#) on the AWS .NET Development blog walks through how to use web identity federation with Facebook and includes code snippets in C# that show how to call `AssumeRoleWithWebIdentity` and how to use the temporary security credentials from that API call to access an S3 bucket.
- The [AWS SDK for iOS](#) and the [AWS SDK for Android](#) contain sample apps. These apps include code that shows how to invoke the identity providers and then how to use the information from these providers to get and use temporary security credentials.
- The article [Web Identity Federation with Mobile Applications](#) discusses web identity federation and shows an example of how to use web identity federation to get access to content in Amazon S3.

About SAML 2.0-based Federation

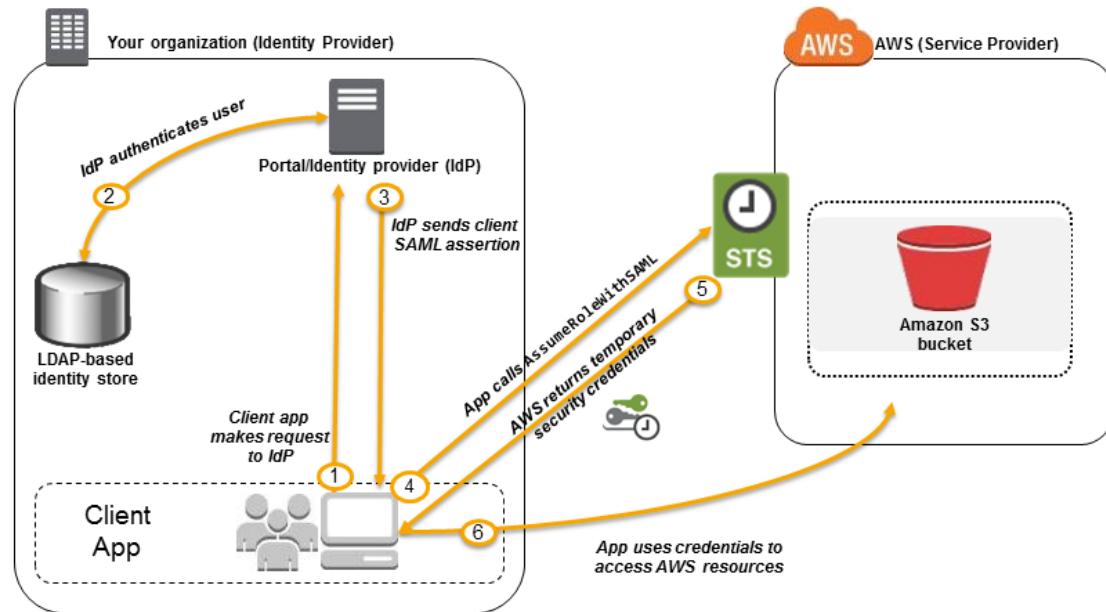
AWS supports identity federation with [SAML 2.0 \(Security Assertion Markup Language 2.0\)](#), an open standard that many identity providers (IdPs) use. This feature enables federated single sign-on (SSO), so users can log into the AWS Management Console or call the AWS APIs without you having to create an IAM user for everyone in your organization. By using SAML, you can simplify the process of configuring federation with AWS, because you can use the IdP's service instead of [writing custom identity proxy code](#).

IAM federation supports these use cases:

- [Federated access to allow a user or application in your organization to call AWS APIs \(p. 150\)](#). You use a SAML assertion (as part of the authentication response) that is generated in your organization to get temporary security credentials. This scenario is similar to other federation scenarios that IAM supports, like those described in [Requesting Temporary Security Credentials \(p. 233\)](#) and [About Web Identity Federation \(p. 143\)](#). However, SAML 2.0-based identity providers in your organization handle many of the details at run time for performing authentication and authorization checking. This is the scenario discussed in this topic.
- [Web-based single sign-on \(SSO\) to the AWS Management Console from your organization \(p. 167\)](#). Users can sign in to a portal in your organization hosted by a SAML 2.0-compatible IdP, select an option to go to AWS, and be redirected to the console without having to provide additional sign-in information. In addition to being able to use a third-party SAML IdP to establish SSO access to the console, you can alternatively create a custom IdP to enable console access for your external users. For more information about building a custom IdP, see [Creating a URL that Enables Federated Users to Access the AWS Management Console \(Custom Federation Broker\) \(p. 169\)](#).

Using SAML-Based Federation for API Access to AWS

Imagine that in your organization, you want to provide a way for users to copy data from their computers to a backup folder. You build an application that users can run on their computers. On the back end, the application reads and writes objects in an S3 bucket. Users don't have direct access to AWS. Instead, the following process is used:



1. A user in your organization uses a client app to request authentication from your organization's IdP.
2. The IdP authenticates the user against your organization's identity store.
3. The IdP constructs a SAML assertion with information about the user and sends the assertion to the client app.
4. The client app makes a request to the IdP.
5. The client app calls the AWS STS `AssumeRoleWithSAML` API, passing the ARN of the SAML provider, the ARN of the role to assume, and the SAML assertion from IdP.
6. The client app uses the temporary security credentials to call Amazon S3 APIs.

Overview of Configuring SAML 2.0-Based Federation

Before you can use SAML 2.0-based federation as described in the preceding scenario and diagram, you must configure your organization's IdP and your AWS account to trust each other. The general process for configuring this trust is described in the following steps. Inside your organization, you must have an [IdP that supports SAML 2.0 \(p. 161\)](#), like Microsoft Active Directory Federation Service (AD FS, part of Windows Server), Shibboleth, or another compatible SAML 2.0 provider.

To configure your organization's IdP and AWS to trust each other

1. You begin by registering AWS with your IdP. In your organization's IdP you register AWS as a service provider (SP) by using the SAML metadata document that you get from the following URL:

`https://signin.aws.amazon.com/static/saml-metadata.xml`
2. Using your organization's IdP, you generate an equivalent metadata XML file that can describe your IdP as an identity provider to AWS. It must include the issuer name, a creation date, an expiration

date, and keys that AWS can use to validate authentication responses (assertions) from your organization.

3. In the IAM console, you create a SAML identity provider entity. As part of this process, you upload the SAML metadata document that was produced by the IdP in your organization in [Step 2](#). For more information, see [Creating SAML Identity Providers \(p. 158\)](#).
4. In IAM, you create one or more IAM roles. In the role's trust policy, you set the SAML provider as the principal, which establishes a trust relationship between your organization and AWS. The role's permission policy establishes what users from your organization are allowed to do in AWS. For more information, see [Creating a Role for a Third-Party Identity Provider \(Federation\) \(p. 194\)](#).
5. In your organization's IdP, you define assertions that map users or groups in your organization to the IAM roles. Note that different users and groups in your organization might map to different IAM roles. The exact steps for performing the mapping depend on what IdP you're using. In the [earlier scenario \(p. 150\)](#) of an Amazon S3 folder for users, it's possible that all users will map to the same role that provides Amazon S3 permissions. For more information, see [Configuring SAML Assertions for the Authentication Response \(p. 163\)](#).

If your IdP enables SSO to the AWS console, then you can configure the maximum duration of the console sessions. For more information, see [Enabling SAML 2.0 Federated Users to Access the AWS Management Console \(p. 167\)](#).

Note

The AWS implementation of SAML 2.0 federation does not support encrypted SAML assertions between the identity provider and AWS. However, the traffic between the customer's systems and AWS is transmitted over an encrypted (TLS) channel.

6. In the application that you're creating, you call the AWS Security Token Service `AssumeRoleWithSAML` API, passing it the ARN of the SAML provider you created in [Step 3](#), the ARN of the role to assume that you created in [Step 4](#), and the SAML assertion about the current user that you get from your IdP. AWS makes sure that the request to assume the role comes from the IdP referenced in the SAML provider.

For more information, see [AssumeRoleWithSAML](#) in the *AWS Security Token Service API Reference*.

7. If the request is successful, the API returns a set of temporary security credentials, which your application can use to make signed requests to AWS. Your application has information about the current user and can access user-specific folders in Amazon S3, as described in the previous scenario.

Overview of the Role to Allow SAML-Federated Access to Your AWS Resources

The role or roles that you create in IAM define what federated users from your organization are allowed to do in AWS. When you create the trust policy for the role, you specify the SAML provider that you created earlier as the `Principal`. You can additionally scope the trust policy with a `Condition` to allow only users that match certain SAML attributes to access the role. For example, you can specify that only users whose SAML affiliation is `staff` (as asserted by <https://openidp.feide.no>) are allowed to access the role, as illustrated by the following sample policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {"Federated": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:saml-provider/ExampleOrgSSOProvider"},
      "Action": "sts:AssumeRoleWithSAML",
      "Condition": {
        "StringEquals": {
          "saml:aud": "https://signin.aws.amazon.com/saml",
          "saml:iss": "https://openidp.feide.no"
        },
        "ForAllValues:StringLike": {"saml:edupersonaffiliation": ["staff"]}
      }
    }
  ]
}
```

```
    }]  
}
```

For more information about the SAML keys that you can check in a policy, see [Available Keys for SAML-Based Federation \(p. 481\)](#).

For the permission policy in the role, you specify permissions as you would for any role. For example, if users from your organization are allowed to administer Amazon Elastic Compute Cloud instances, you must explicitly allow Amazon EC2 actions in the permissions policy, such as those in the [AmazonEC2FullAccess](#) managed policy.

Uniquely Identifying Users in SAML-Based Federation

When you create access policies in IAM, it's often useful to be able to specify permissions based on the identity of users. For example, for users who have been federated using SAML, an application might want to keep information in Amazon S3 using a structure like this:

```
myBucket/app1/user1  
myBucket/app1/user2  
myBucket/app1/user3
```

You can create the bucket (`myBucket`) and folder (`app1`) through the Amazon S3 console or the AWS CLI, since those are static values. However, the user-specific folders (`user1`, `user2`, `user3`, etc.) have to be created at run time using code, since the value that identifies the user isn't known until the first time the user signs in through the federation process.

To write policies that reference user-specific details as part of a resource name, the user identity has to be available in SAML keys that can be used in policy conditions. The following keys are available for SAML 2.0-based federation for use in IAM policies. You can use the values returned by the following keys to create unique user identifiers for resources like Amazon S3 folders.

- `saml:namequalifier`. A hash value based on the concatenation of the `Issuer` response value (`saml:iss`) and a string with the AWS account ID and the friendly name (the last part of the ARN) of the SAML provider in IAM. The concatenation of the account ID and friendly name of the SAML provider is available to IAM policies as the key `saml:doc`. The account ID and provider name must be separated by a '/' as in "123456789012/provider_name". For more information, see the `saml:doc` key at [Available Keys for SAML-Based Federation \(p. 481\)](#).

The combination of `NameQualifier` and `Subject` can be used to uniquely identify a federated user. The following pseudocode shows how this value is calculated. In this pseudocode `+ indicates concatenation`, `SHA1` represents a function that produces a message digest using SHA-1, and `Base64` represents a function that produces Base-64 encoded version of the hash output.

```
Base64 ( SHA1 ( "https://example.com/saml" + "123456789012" + "/  
MySAMLIdP" ) )
```

For more information about the policy keys that are available for SAML-based federation, see [Available Keys for SAML-Based Federation \(p. 481\)](#).

- `saml:sub` (string). This is the subject of the claim, which includes a value that uniquely identifies an individual user within an organization (for example, `_ccb88bf52c2510eabe00c1642d4643f41430fe25e3`).
- `saml:sub_type` (string). This key can be `persistent`, `transient`, or the full `Format` URI from the `Subject` and `NameID` elements used in your SAML assertion. A value of `persistent` indicates that the value in `saml:sub` is the same for a user across all sessions. If the value is `transient`, the user has a different `saml:sub` value for each session. For information about the `NameID` element's `Format` attribute, see [Configuring SAML Assertions for the Authentication Response \(p. 163\)](#).

The following example shows a permission policy that uses the preceding keys to grant permissions to a user-specific folder in Amazon S3. The policy assumes that the Amazon S3 objects are identified using a prefix that includes both `saml:namequalifier` and `saml:sub`. Notice that the `Condition` element includes a test to be sure that `saml:sub_type` is set to `persistent`. If it is set to `transient`, the `saml:sub` value for the user can be different for each session, and the combination of values should not be used to identify user-specific folders.

```
>{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:PutObject",
      "s3:DeleteObject"
    ],
    "Resource": [
      "arn:aws:s3:::exampleorgBucket/backup/${saml:namequalifier}/${saml:sub}",
      "arn:aws:s3:::exampleorgBucket/backup/${saml:namequalifier}/${saml:sub}/*"
    ],
    "Condition": {"StringEquals": {"saml:sub_type": "persistent"}}
  }
}
```

For more information about mapping assertions from the IdP to policy keys, see [Configuring SAML Assertions for the Authentication Response \(p. 163\)](#).

Creating IAM Identity Providers

When you want to configure federation with an external identity provider service (IdP), you create an *identity provider* in the IAM console to inform AWS about the IdP and its configuration. This establishes "trust" between your AWS account and the IdP. The following topics include details about how to create an identity provider in IAM for each of the IdP types.

Topics

- [Creating OpenID Connect \(OIDC\) Identity Providers \(p. 153\)](#)
- [Creating SAML Identity Providers \(p. 158\)](#)

Creating OpenID Connect (OIDC) Identity Providers

OIDC identity providers are entities in IAM that describe an identity provider (IdP) service that supports the [OpenID Connect](#) (OIDC) standard. You use an OIDC identity provider when you want to establish trust between an OIDC-compatible IdP—such as Google, Salesforce, and many others—and your AWS account. This is useful if you are creating a mobile app or web application that requires access to AWS resources, but you don't want to create custom sign-in code or manage your own user identities. For more information about this scenario, see the section called ["About Web Identity Federation" \(p. 143\)](#).

You can create and manage an OIDC identity provider using the AWS Management Console, the AWS Command Line Interface, the Tools for Windows PowerShell, or the IAM API.

Topics

- [Creating and Managing an OIDC Provider \(AWS Management Console\) \(p. 154\)](#)
- [Creating and Managing an OIDC Identity Provider \(AWS CLI, Tools for Windows PowerShell, and IAM API\) \(p. 155\)](#)
- [Obtaining the Thumbprint for an OpenID Connect Identity Provider \(p. 155\)](#)

Creating and Managing an OIDC Provider (AWS Management Console)

Follow these instructions to create and manage an OIDC provider in the AWS Management Console.

To create an OIDC identity provider

1. Before you create an OIDC identity provider in IAM, you must register your application with the IdP to receive a *client ID*. The client ID (also known as *audience*) is a unique identifier for your app that is issued to you when you register your app with the IdP. For more information about obtaining a client ID, see the documentation for your IdP.
2. Open the IAM console at <https://console.aws.amazon.com/iam/>.
3. In the navigation pane, click **Identity Providers**, and then click **Create Provider**.
4. For **Provider Type**, click **Choose a provider type**, and then choose **OpenID Connect**.
5. For **Provider URL**, type the URL of the IdP. The URL must comply with these restrictions:
 - The URL is case-sensitive.
 - The URL must begin with **https://**
 - The URL cannot include a colon (:) character, and therefore cannot specify a port number. This means that the server must be listening on the default port 443.
 - Within your AWS account, each OIDC identity provider must use a unique URL.
6. For **Audience**, type the client ID of the application that you registered with the IdP and received in [Step 1](#), and that will make requests to AWS. If you have additional client IDs (also known as *audiences*) for this IdP, you can add them later on the provider detail page. Click **Next Step**.
7. Use the **Thumbprint** to verify the server certificate of your IdP. To learn how, see [Obtaining the Thumbprint for an OpenID Connect Identity Provider \(p. 155\)](#). Click **Create**.
8. In the confirmation message at the top of the screen, click **Do this now** to go to the **Roles** tab to create a role for this identity provider. For more information about creating a role for an OIDC identity provider, see [Creating a Role for a Third-Party Identity Provider \(Federation\) \(p. 194\)](#). OIDC identity providers must have a role in order to access your AWS account. To skip this step and create the role later, click **Close**.

To add or remove a thumbprint or client ID (also known as audience) for an OIDC identity provider

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, click **Identity Providers**, then click the name of the identity provider that you want to update.
3. To add a thumbprint or audience, click **Add a Thumbprint** or **Add an Audience**. To remove a thumbprint or audience, click **Remove** next to the item that you want to remove.

Note

An OIDC identity provider must have at least 1 and can have a maximum of 5 thumbprints.
An OIDC identity provider must have at least 1 and can have a maximum of 100 audiences.

When you are done, click **Save Changes**.

To delete an OIDC identity provider

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, click **Identity Providers**.
3. Select the check box next to the identity provider that you want to delete.
4. Click **Delete Providers**.

Creating and Managing an OIDC Identity Provider (AWS CLI, Tools for Windows PowerShell, and IAM API)

Use the following commands to create and manage an OIDC provider.

To create a new OIDC provider

- AWS CLI: `aws iam create-open-id-connect-provider`
- Tools for Windows PowerShell: `New-IAMOpenIDConnectProvider`
- IAM API: `CreateOpenIDConnectProvider`

To add a new client ID to an existing OIDC provider

- AWS CLI: `aws iam add-client-id-to-open-id-connect-provider`
- Tools for Windows PowerShell: `Add-IAMClientIDToOpenIDConnectProvider`
- IAM API: `AddClientIDToOpenIDConnectProvider`

To remove a client ID from an existing OIDC provider

- AWS CLI: `aws iam remove-client-id-from-open-id-connect-provider`
- Tools for Windows PowerShell: `Remove-IAMClientIDToOpenIDConnectProvider`
- IAM API: `RemoveClientIDFromOpenIDConnectProvider`

To update the list of server certificate thumbprints for an existing OIDC provider

- AWS CLI: `aws iam update-open-id-connect-provider-thumbprint`
- Tools for Windows PowerShell: `Update-IAMOpenIDConnectProviderThumbprint`
- IAM API: `UpdateOpenIDConnectProviderThumbprint`

To get a list of all the OIDC providers in your AWS account

- AWS CLI: `aws iam list-open-id-connect-providers`
- Tools for Windows PowerShell: `Get-IAMOpenIDConnectProviders`
- IAM API: `ListOpenIDConnectProviders`

To get detailed information about an OIDC provider

- AWS CLI: `aws iam get-open-id-connect-provider`
- Tools for Windows PowerShell: `Get-IAMOpenIDConnectProvider`
- IAM API: `GetOpenIDConnectProvider`

To delete an OIDC provider

- AWS CLI: `aws iam delete-open-id-connect-provider`
- Tools for Windows PowerShell: `Remove-IAMOpenIDConnectProvider`
- IAM API: `DeleteOpenIDConnectProvider`

Obtaining the Thumbprint for an OpenID Connect Identity Provider

When you [create an OpenID Connect \(OIDC\) identity provider \(p. 153\)](#) in IAM, you must supply a thumbprint for the identity provider (IdP). The thumbprint is a signature for the unique server certificate

that is used by the OIDC-compatible IdP. When you create an OIDC identity provider in IAM, you are trusting identities authenticated by that IdP with access to your AWS account. By supplying the OIDC IdP's thumbprint, you assert to AWS that you wish to trust a particular OIDC IdP with this access.

When you create an OIDC identity provider with [the AWS Command Line Interface, the Tools for Windows PowerShell, or the IAM API \(p. 155\)](#), you must obtain the thumbprint manually and supply it to AWS. When you create an OIDC identity provider with [the IAM console \(p. 153\)](#), the console attempts to fetch the thumbprint for you. We recommend that you also obtain the thumbprint for your OIDC IdP manually and verify that the thumbprint obtained by the IAM console matches the one you expect for your OIDC provider.

You use a web browser and the OpenSSL command line tool to obtain the thumbprint for an OIDC provider. For more information, see the following sections.

To obtain the thumbprint for an OIDC IdP

1. Before you can obtain the thumbprint for an OIDC IdP, you need to obtain the OpenSSL command-line tool. You use this tool to download the OIDC IdP's certificate chain and produce a thumbprint of the final certificate in the certificate chain. If you need to install and configure OpenSSL, follow the instructions at [Install OpenSSL \(p. 157\)](#) and [Configure OpenSSL \(p. 158\)](#).
2. Start with the OIDC IdP's URL (for example, `https://server.example.com`), and then add `.well-known/openid-configuration` to form the URL for the IdP's configuration document, like the following:

`https://server.example.com/.well-known/openid-configuration`

Open this URL in a web browser, replacing `server.example.com` with your IdP's server name.

3. In the document displayed in your web browser, find "jwks_uri". (Use your web browser's **Find** feature to locate this text on the page.) Immediately following the text "jwks_uri" you will see a colon (:) followed by a URL. Copy the fully qualified domain name of the URL. Do not include the `https://` or any path that comes after the top-level domain.
4. Use the OpenSSL command line tool to execute the following command. Replace `keys.example.com` with the domain name you obtained in [Step 3](#).

```
openssl s_client -showcerts -connect keys.example.com:443
```

5. In your command window, scroll up until you see a certificate similar to the following example. If you see more than one certificate, find the last certificate that is displayed (at the bottom of the command output).

```
-----BEGIN CERTIFICATE-----
MIICitCCAfICCCCD6m70Rw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxszAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC01BTSBDb25zb2x1MRIwEAYDVQQDEwlUZXN0Q21sYWMxHzAd
BgkqhkiG9w0BCQEWEg5vb25lQGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxszAJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC01BTSBDb25z
b2x1MRIwEAYDVQQDEwlUZXN0Q21sYWMxHzAdBgkqhkiG9w0BCQEWEg5vb25lQGft
YXpvbi5jb20wgZ8wDQYJKoZIhvCNQEBBQADgY0AMIGJAOGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLYgV1k60CpiwsZ3G93vUEIO3IyNoh/f0wYK8m9T
rDHudUZg3qX4walG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzzSwY6786m86gpE
Ibb3OhjZnzcvQAArHhd1QWIIMm2nrAgMBAAEwDQYJKoZIhvCNQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtsS475iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJ10ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----
```

Copy the certificate (including the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- lines) and paste it into a text file. Then save the file with the file name **certificate.crt**.

6. Use the OpenSSL command-line tool to execute the following command.

```
openssl x509 -in certificate.crt -fingerprint -noout
```

Your command window displays the certificate thumbprint, which looks similar to the following example:

```
SHA1 Fingerprint=99:0F:41:93:97:2F:2B:EC:F1:2D:DE:DA:52:37:F9:C9:52:F2:0D:9E
```

Remove the colon characters (:) from this string to produce the final thumbprint, like this:

```
990F4193972F2BECF12DDEDA5237F9C952F20D9E
```

7. If you are creating the IAM identity provider with the AWS CLI, Tools for Windows PowerShell, or the IAM API, supply this thumbprint when creating the provider.

If you are creating the OIDC provider in the IAM console, compare this thumbprint to the thumbprint that you see in the console on the **Verify Provider Information** page when creating an OIDC provider.

Important

If the thumbprint you obtained does not match the one you see in the console, you should not create the OIDC provider in IAM. Instead, you should wait a while and then try again to create the OIDC provider, ensuring that the thumbprints match before you create the provider. If the thumbprints still do not match after a second attempt, use the [IAM Forum](#) to contact AWS.

Install OpenSSL

If you don't already have OpenSSL installed, follow the instructions in this section.

To install OpenSSL on Linux or Unix

1. Go to [OpenSSL: Source, Tarballs](#) (<https://openssl.org/source/>).
2. Download the latest source and build the package.

To install OpenSSL on Windows

1. Go to [OpenSSL: Binary Distributions](#) (<https://wiki.openssl.org/index.php/Binaries>) for a list of sites from which you can install the Windows version.
2. Follow the instructions on your selected site to start the installation.
3. If you are asked to install the **Microsoft Visual C++ 2008 Redistributables** and it is not already installed on your system, choose the download link appropriate for your environment. Follow the instructions provided by the **Microsoft Visual C++ 2008 Redistributable Setup Wizard**.

Note

If you are not sure whether the Microsoft Visual C++ 2008 Redistributables is already installed on your system, you can try installing OpenSSL first. The OpenSSL installer displays an alert if the Microsoft Visual C++ 2008 Redistributables is not yet installed. Make sure you install the architecture (32-bit or 64-bit) that matches the version of OpenSSL that you install.

4. After you have installed the Microsoft Visual C++ 2008 Redistributables, select the appropriate version of the OpenSSL binaries for your environment and save the file locally. Start the **OpenSSL Setup Wizard**.
5. Follow the instructions described in the **OpenSSL Setup Wizard**.

Configure OpenSSL

Before you use OpenSSL commands, you must configure the operating system so that it has information about the location where OpenSSL is installed.

To configure OpenSSL on Linux or Unix

1. At the command line, set the `OpenSSL_HOME` variable to the location of the OpenSSL installation:

```
$ export OpenSSL_HOME=path_to_your_OpenSSL_installation
```

2. Set the path to include the OpenSSL installation:

```
$ export PATH=$PATH:$OpenSSL_HOME/bin
```

Note

Any changes you make to environment variables with the `export` command are valid only for the current session. You can make persistent changes to the environment variables by setting them in your shell configuration file. For more information, see the documentation for your operating system.

To configure OpenSSL on Windows

1. Open a **Command Prompt** window.
2. Set the `OpenSSL_HOME` variable to the location of the OpenSSL installation:

```
C:\> set OpenSSL_HOME=path_to_your_OpenSSL_installation
```

3. Set the `OpenSSL_CONF` variable to the location of the configuration file in your OpenSSL installation:

```
C:\> set OpenSSL_CONF=path_to_your_OpenSSL_installation\bin\openssl.cfg
```

4. Set the path to include the OpenSSL installation:

```
C:\> set Path=%Path%;%OpenSSL_HOME%\bin
```

Note

Any changes you make to Windows environment variables in a **Command Prompt** window are valid only for the current command line session. You can make persistent changes to the environment variables by setting them as system properties. The exact procedures depends on what version of Windows you're using. (For example, in Windows 7, open **Control Panel**, **System and Security**, **System**. Then choose **Advanced system settings**, **Advanced tab**, **Environment Variables**.) For more information, see the Windows documentation.

Creating SAML Identity Providers

A SAML 2.0 identity provider is an entity in IAM that describes an identity provider (IdP) service that supports the [SAML 2.0 \(Security Assertion Markup Language 2.0\)](#) standard. You use a SAML identity

provider when you want to establish trust between an SAML-compatible IdP such as Shibboleth or Active Directory Federation Services so that users in your organization can access AWS resources. SAML identity providers in IAM are used as principals in an IAM trust policy.

For more information about this scenario, see [About SAML 2.0-based Federation \(p. 149\)](#).

You can create and manage a SAML identity provider in the AWS Management Console or with AWS CLI, Tools for Windows PowerShell, or AWS API calls.

After you create a SAML provider, you must create one or more IAM roles. A role is an identity in AWS that doesn't have its own credentials (as a user does) but is, in this context, dynamically assigned to a federated user that is authenticated by your organization's identity provider (IdP). The role permits your organization's IdP to request temporary security credentials for access to AWS. The policies assigned to the role determine what the federated users are allowed to do in AWS. To create a role for SAML federation, see [Creating a Role for a Third-Party Identity Provider \(Federation\) \(p. 194\)](#).

Finally, after you create the role, you complete the SAML trust by configuring your IdP with information about AWS and the role(s) that you want your federated users to use. This is referred to as configuring relying party trust between your IdP and AWS. To configure relying party trust, see [Configuring your SAML 2.0 IdP with Relying Party Trust and Adding Claims \(p. 160\)](#).

Topics

- [Creating and Managing a SAML Identity Provider \(AWS Management Console\) \(p. 159\)](#)
- [Managing a SAML Provider \(AWS CLI, Tools for Windows PowerShell and AWS API\) \(p. 160\)](#)
- [Configuring your SAML 2.0 IdP with Relying Party Trust and Adding Claims \(p. 160\)](#)
- [Integrating Third-Party SAML Solution Providers with AWS \(p. 161\)](#)
- [Configuring SAML Assertions for the Authentication Response \(p. 163\)](#)

[Creating and Managing a SAML Identity Provider \(AWS Management Console\)](#)

You can use the AWS Management Console to create and delete SAML identity providers.

To create a SAML identity provider

1. Before you can create a SAML identity provider, you need the SAML metadata document that you get from the IdP that includes the issuer's name, expiration information, and keys that can be used to validate the SAML authentication response (assertions) that are received from the IdP. To generate the metadata document, use the identity management software your organization uses as its IdP. For instructions on how to configure many of the available IdPs to work with AWS, including how to generate the required SAML metadata document, see [Integrating Third-Party SAML Solution Providers with AWS \(p. 161\)](#).

Important

The metadata file must be encoded in UTF-8 format without a byte order mark (BOM). Also, the x.509 certificate that is included as part of the SAML metadata document must use a key size of at least 1024 bits. If the key size is smaller, the IdP creation fails with an "Unable to parse metadata" error. To remove the BOM, you can encode the file as UTF-8 using a text editing tool, such as Notepad++.

2. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
3. In the navigation pane, click **Identity Providers** and then click **Create Provider**.
4. For **Provider Type**, click **Choose a provider type** and click **SAML**.
5. Type a name for the identity provider.
6. For **Metadata Document**, click **Choose File**, specify the SAML metadata document that you downloaded in Step 1, and click **Open**. Click **Next Step**.
7. Verify the information you have provided, and click **Create**.

To delete a SAML provider

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, click **Identity Providers**.
3. Select the check box next to the identity provider that you want to delete.
4. Click **Delete Providers**.

Managing a SAML Provider (AWS CLI, Tools for Windows PowerShell and AWS API)

Use the following commands to create and manage a SAML provider.

To create an identity provider and upload a metadata document

- AWS CLI: `aws iam create-saml-provider`
- Tools for Windows PowerShell: `New-IAMSAMLProvider`
- AWS API: `CreateSAMLProvider`

To upload a new metadata document for an IdP

- AWS CLI: `aws iam update-saml-provider`
- Tools for Windows PowerShell: `Update-IAMSAMLProvider`
- AWS API: `UpdateSAMLProvider`

To get information about a specific provider, such as the ARN, creation date, and expiration

- AWS CLI: `aws iam get-saml-provider`
- Tools for Windows PowerShell: `Get-IAMSAMLProvider`
- AWS API: `GetSAMLProvider`

To list information for all IdPs, such as the ARN, creation date, and expiration

- AWS CLI: `aws iam list-saml-providers`
- Tools for Windows PowerShell: `Get-IAMSAMLProviders`
- AWS API: `ListSAMLProviders`

To delete an IdP

- AWS CLI: `aws iam delete-saml-provider`
- Tools for Windows PowerShell: `Remove-IAMSAMLProvider`
- AWS API: `DeleteSAMLProvider`

Configuring your SAML 2.0 IdP with Relying Party Trust and Adding Claims

When you create a SAML provider and the role for SAML access in IAM, you are essentially telling AWS about the identity provider (IdP) and what its users are allowed to do. Your next step is to then tell the IdP about AWS as a service provider. This is called adding *relying party trust* between your IdP and AWS. The exact process for adding relying party trust depends on what IdP you're using; for details, see the documentation for your identity management software.

Many IdPs allow you to specify a URL from which the IdP can read an XML document that contains relying party information and certificates. For AWS, you can use <https://signin.aws.amazon.com/static/saml-metadata.xml>

If you can't specify a URL directly, then download the XML document from the preceding URL and import it into your IdP software.

You also need to create appropriate claim rules in your IdP that specify AWS as a relying party. When the IdP sends a SAML response to the AWS endpoint, it includes a SAML *assertion* that contains one or more *claims*. A claim is information about the user and its groups. A claim rule maps that information into SAML attributes. This lets you make sure that SAML authentication responses from your IdP contain the necessary attributes that AWS uses in IAM policies to check permissions for federated users. For more information, see the following topics:

- [Overview of the Role to Allow SAML-Federated Access to Your AWS Resources \(p. 151\)](#). This topic discusses using SAML-specific keys in IAM policies and how to use them to restrict permissions for SAML-federated users.
- [Configuring SAML Assertions for the Authentication Response \(p. 163\)](#). This topic discusses how to configure SAML claims that include information about the user. The claims are bundled into a SAML assertion and included in the SAML response that is sent to AWS. You must ensure that the information needed by AWS policies is included in the SAML assertion in a form that AWS can recognize and use.
- [Integrating Third-Party SAML Solution Providers with AWS \(p. 161\)](#). This topic provides links to documentation provided by third-party organizations about how to integrate identity solutions with AWS.

Integrating Third-Party SAML Solution Providers with AWS

The following links help you configure third-party SAML 2.0 identity provider solutions to work with AWS federation.

Solution	More information
Auth0	AWS Integration in Auth0 – This page on the Auth0 documentation website describes how to set up single sign-on (SSO) with the AWS Management Console and includes a JavaScript example.
Bitium	Configuring SAML for Amazon Web Services (AWS) – This article on the Bitium support site explains how to use Bitium to set up AWS with SAML SSO.
Centrify	Configure Centrify and Use SAML for SSO to AWS – This page on the Centrify website explains how to configure Centrify to use SAML for SSO to AWS.
CertiVox	Setting up M-Pin SSO as an Identity Provider within AWS – This page on the CertiVox website explains how to configure an AWS service provider for SSO authentication through your M-Pin SSO system.
Clearlogin	Amazon Web Services Setup – This article in the Clearlogin Help Center explains how to set up SSO functionality between Clearlogin and AWS.
Google G Suite	Amazon Web Services cloud application – This article on the Google G Suite Administrator Help site describes how to

Solution	More information
	configure G Suite as a SAML 2.0 IdP with AWS as the service provider.
Identacor	Configuring SSO (SAML) for AWS – This article on the Identacor website describes how to set up and enable SSO for AWS.
Matrix42	MyWorkspace Getting Started Guide – This guide describes how to integrate AWS identity services with Matrix42 MyWorkspace.
Microsoft Active Directory Federation Services (AD FS)	<p>Enabling Federation to AWS Using Windows Active Directory, AD FS, and SAML 2.0 – This post on the AWS Security Blog shows how to set up AD FS on an EC2 instance and enable SAML federation with AWS.</p> <p>PowerShell Automation to Give AWS Console Access – This post on Sivaprasad Padisetty's blog describes how to use Windows PowerShell to automate the process of setting up Active Directory and AD FS as well as enabling SAML federation with AWS.</p>
miniOrange	SSO for AWS – This page on the miniOrange website describes how to establish secure access to AWS for enterprises and full control over access of AWS applications.
Okta	Integrating the Amazon Web Services Command Line Interface Using Okta – From this page on the Okta support site you can learn how to configure Okta for use with AWS.
OneLogin	Configuring SAML Single-Role for AWS – This article in the OneLogin Help Center explains how to set up SSO functionality between OneLogin and AWS.
Ping Identity	<p>PingFederate AWS Connector – From this page on the Ping Identity website, you can download a PDF file that describes how to configure a PingFederate server to enable SSO for user accounts with AWS.</p> <p>Configuring an SSO connection to AWS – This Ping Identity page describes how to configure an SSO connection from PingOne to AWS.</p>
RadiantLogic	Radiant Logic Technology Partners – Radiant Logic's RadiantOne Federated Identity Service integrates with AWS to provide an identity hub for SAML-based SSO.
Salesforce.com	How to configure SSO from Salesforce to AWS – This how-to article on the Salesforce.com developer site describes how to set up an identity provider (IdP) in Salesforce and configure AWS as a service provider.
SecureAuth	AWS - SecureAuth SAML SSO – This article on the SecureAuth website describes how to set up SAML integration with AWS for a SecureAuth appliance.

Solution	More information
Shibboleth	How to Use Shibboleth for SSO to the AWS Management Console – This entry on the AWS Security Blog provides a step-by-step tutorial on how to set up Shibboleth and configure it as an identity provider for AWS.

For more details, see the [IAM Partners](#) page on the AWS website.

Configuring SAML Assertions for the Authentication Response

In your organization, after a user's identity has been verified, the IdP sends an authentication response to the AWS SAML endpoint at <https://signin.aws.amazon.com/saml>. This response is a POST request that includes a SAML token that adheres to the [HTTP POST Binding for SAML 2.0](#) standard and that contains the following elements, or *claims*. You configure these claims in your SAML-compatible IdP. Refer to the documentation for your IdP for instructions on how to enter these claims.

When the IdP sends the response containing the claims to AWS, many of the incoming claims map to AWS context keys that can be checked in IAM policies using the `Condition` element. A listing of the available mappings follows in the section [Mapping SAML Attributes to AWS Trust Policy Context Keys \(p. 165\)](#).

Subject and NameID

The following excerpt shows an example. Substitute your own values for the marked ones. There must be exactly 1 `SubjectConfirmation` element with a `SubjectConfirmationData` element that includes both the `NotOnOrAfter` attribute and a `Recipient` attribute with a value that must match the AWS endpoint (<https://signin.aws.amazon.com/saml>), as shown in the following example. For information about the name identifier formats supported for single sign-on interactions, see [Oracle Sun OpenSSO Enterprise Administration Reference](#).

```
<Subject>
  <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent">cbb88bf52c2510eabe00c1642d4643f41430fe25e3</NameID>
  <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
    <SubjectConfirmationData NotOnOrAfter="2013-11-05T02:06:42.876Z">
      Recipient="https://signin.aws.amazon.com/saml">
    </SubjectConfirmation>
  </Subject>
```

AudienceRestriction and Audience

For security reasons, AWS must be included as an audience in the SAML assertion your IdP sends to AWS. Therefore, the value of the `Audience` element should match one of the following two values, either <https://signin.aws.amazon.com/saml> or <urn:amazon:webservices>. The following sample XML snippets from SAML assertions show how this key can be specified by the IdP. Include whichever sample applies to your use case.

```
<Conditions>
  <AudienceRestriction>
    <Audience>https://signin.aws.amazon.com/saml</Audience>
  </AudienceRestriction>
</Conditions>
```

```
<Conditions>
  <AudienceRestriction>
    <Audience>urn:amazon:webservices</Audience>
  </AudienceRestriction>
</Conditions>
```

```
</Conditions>
```

Important

The SAML AudienceRestriction value in the SAML assertion from the IdP does *not* map to the saml:aud context key that you can test in an IAM policy. Instead, the saml:aud context key comes from the SAML *recipient* attribute because it is the SAML equivalent to the OIDC audience field, for example, by accounts.google.com:aud.

An Attribute element with the Name attribute set to <https://aws.amazon.com/SAML/Attributes/Role>

This element contains one or more AttributeValue elements that list the IAM role and SAML identity provider to which the user is mapped by your IdP. The role and identity provider are specified as a comma-delimited pair of ARNs in the same format as the RoleArn and PrincipalArn parameters that are passed to [AssumeRoleWithSAML](#). This element must contain at least one role-provider pair—that is, at least one AttributeValue element—and can contain multiple pairs. If the element contains multiple pairs, then the user is asked to select which role to assume when he or she uses WebSSO to sign into the AWS Management Console.

Important

The value of the Name attribute in the Attribute tag is case-sensitive. It must be set to <https://aws.amazon.com/SAML/Attributes/Role> exactly.

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/Role">
  <AttributeValue>arn:aws:iam::account-number:role/role-name1,arn:aws:iam::account-
  number:saml-provider/provider-name</AttributeValue>
  <AttributeValue>arn:aws:iam::account-number:role/role-name2,arn:aws:iam::account-
  number:saml-provider/provider-name</AttributeValue>
  <AttributeValue>arn:aws:iam::account-number:role/role-name3,arn:aws:iam::account-
  number:saml-provider/provider-name</AttributeValue>
</Attribute>
```

An Attribute element with the Name attribute set to <https://aws.amazon.com/SAML/Attributes/RoleSessionName>

This element contains one AttributeValue element that provides an identifier for the AWS temporary credentials that are issued for SSO and is used to display user information in the AWS Management Console. The value in the AttributeValue element must be between 2 and 64 characters long, can contain only alphanumeric characters, underscores, and the following characters: + (plus sign), = (equals sign), , (comma), . (period), @ (at symbol), and - (hyphen). It cannot contain spaces. The value is typically a user ID (bobsmith) or an email address (bobsmith@example.com). It should not be a value that includes a space, like a user's display name (Bob Smith).

Important

The value of the Name attribute in the Attribute tag is case-sensitive. It must be set to <https://aws.amazon.com/SAML/Attributes/RoleSessionName> exactly.

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/RoleSessionName">
  <AttributeValue>user-id-name</AttributeValue>
</Attribute>
```

An optional Attribute element with the SessionDuration attribute set to <https://aws.amazon.com/SAML/Attributes/SessionDuration>

This element contains one AttributeValue element that specifies the maximum time that the user can access the AWS Management Console before having to request new temporary credentials. The value is an integer representing the number of seconds, and can be a maximum of 43200 seconds (12 hours). If this attribute is not present, then the maximum session duration defaults to one hour (the default value of the DurationSeconds parameter of the [AssumeRoleWithSAML](#)

API). To use this attribute, you must configure the SAML provider to provide single sign-on access to the AWS Management Console through the console sign-in web endpoint at <https://signin.aws.amazon.com/saml>. Note that this attribute extends sessions only to the AWS Management Console. It cannot extend the lifetime of other credentials. However, if it is present in an `AssumeRoleWithSAML` API call, it can be used to *shorten* the lifetime of the credentials returned by the call to less than the default of 60 minutes.

Note, too, that if a `SessionNotOnOrAfter` attribute is also defined, then the *lesser* value of the two attributes, `SessionDuration` or `SessionNotOnOrAfter`, establishes the maximum duration of the console session.

When you enable console sessions with an extended duration the risk of compromise of the credentials rises. To help you mitigate this risk, you can immediately disable the active console sessions for any role by choosing **Revoke Sessions** on the **Role Summary** page in the IAM console. For more information, see [Revoking IAM Role Temporary Security Credentials \(p. 220\)](#).

Important

The value of the `Name` attribute in the `Attribute` tag is case-sensitive. It must be set to <https://aws.amazon.com/SAML/Attributes/SessionDuration> exactly.

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SessionDuration">
  <AttributeValue>7200</AttributeValue>
</Attribute>
```

Mapping SAML Attributes to AWS Trust Policy Context Keys

The tables in this section list commonly used SAML attributes and how they map to trust policy condition context keys in AWS. You can use these keys to control access to a role by evaluating them against the values included in the assertions that accompany a SAML request to access a role.

Important

These keys are available only in IAM trust policies (policies that determine who can assume a role) and are not applicable to permissions policies.

In the `eduPerson` and `eduOrg` attributes table, values are typed either as strings or as lists of strings. For string values, you can test these values in IAM trust policies using `StringEquals` or `StringLike` conditions. For values that contain a list of strings, you can use the `ForAnyValue` and `ForAllValues` policy set operators (p. 447) to test the values in trust policies.

Note

You should include only one claim per AWS context key. If you include more than one, only one claim will be mapped.

eduPerson and eduOrg Attributes

eduPerson or eduOrg attribute (<code>Name</code> key)	Maps to this AWS context key (<code>FriendlyName</code> key)	Type
<code>urn:oid:1.3.6.1.4.1.5923.1.1.1.1</code>	<code>eduPersonAffiliation</code>	List of strings
<code>urn:oid:1.3.6.1.4.1.5923.1.1.1.2</code>	<code>eduPersonNickname</code>	List of strings
<code>urn:oid:1.3.6.1.4.1.5923.1.1.1.3</code>	<code>eduPersonOrgDN</code>	String
<code>urn:oid:1.3.6.1.4.1.5923.1.1.1.4</code>	<code>eduPersonOrgUnitDN</code>	List of strings
<code>urn:oid:1.3.6.1.4.1.5923.1.1.1.5</code>	<code>eduPersonPrimaryAffiliation</code>	String
<code>urn:oid:1.3.6.1.4.1.5923.1.1.1.6</code>	<code>eduPersonPrincipalName</code>	String
<code>urn:oid:1.3.6.1.4.1.5923.1.1.1.7</code>	<code>eduPersonEntitlement</code>	List of strings

eduPerson or eduOrg attribute (Name key)	Maps to this AWS context key (FriendlyName key)	Type
urn:oid:1.3.6.1.4.1.5923.1.1.1.8	eduPersonPrimaryOrgUnitDN	String
urn:oid:1.3.6.1.4.1.5923.1.1.1.9	eduPersonScopedAffiliation	List of strings
urn:oid:1.3.6.1.4.1.5923.1.1.1.10	eduPersonTargetedID	List of strings
urn:oid:1.3.6.1.4.1.5923.1.1.1.11	eduPersonAssurance	List of strings
urn:oid:1.3.6.1.4.1.5923.1.2.1.2	eduOrgHomePageURI	List of strings
urn:oid:1.3.6.1.4.1.5923.1.2.1.3	eduOrgIdentityAuthNPolicy	List of strings
urn:oid:1.3.6.1.4.1.5923.1.2.1.4	eduOrgLegalName	List of strings
urn:oid:1.3.6.1.4.1.5923.1.2.1.5	eduOrgSuperiorURI	List of strings
urn:oid:1.3.6.1.4.1.5923.1.2.1.6	eduOrgWhitePagesURI	List of strings
urn:oid:2.5.4.3	cn	List of strings

Active Directory Attributes

AD attribute	Maps to this AWS context key	Type
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name	name	String
http://schemas.xmlsoap.org/claims/CommonName	commonName	String
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname	givenName	String
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname	surname	String
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress	mail	String
http://schemas.microsoft.com/ws/2008/06/identity/claims/primarygroupsid	uid	String

X.500 Attributes

X.500 Attribute	Maps to this AWS context key	Type
2.5.4.3	commonName	String
2.5.4.4	surname	String
2.4.5.42	givenName	String
2.5.4.45	x500UniqueIdentifier	String
0.9.2342.19200300100.1.1	uid	String

X.500 Attribute	Maps to this AWS context key	Type
0.9.2342.19200300100.1.3	mail	String
0.9.2342.19200300.100.1.45	organizationStatus	String

Enabling SAML 2.0 Federated Users to Access the AWS Management Console

You can use a role to configure your SAML 2.0-compliant IdP and AWS to permit your federated users to access the AWS Management Console. The role grants the user permissions to carry out tasks in the console. If instead you want to give SAML federated users other ways to access AWS, see one of these topics:

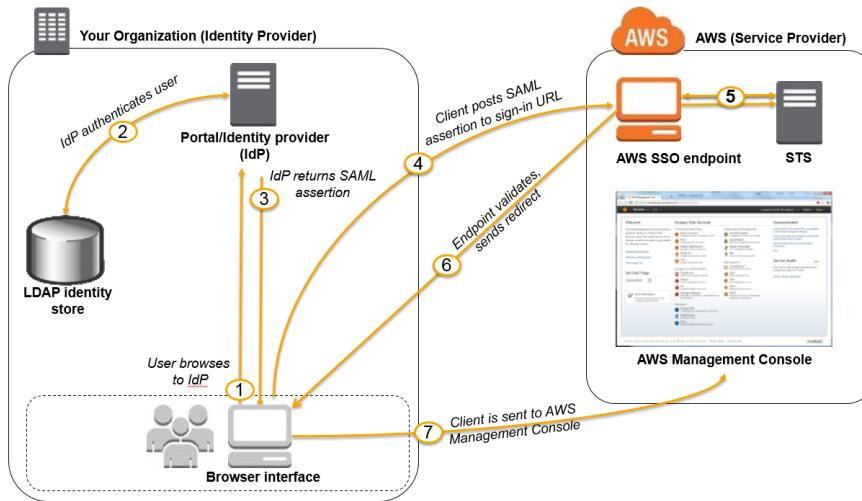
- AWS CLI: [Switching to an IAM Role \(AWS Command Line Interface\) \(p. 211\)](#)
- Tools for Windows PowerShell: [Switching to an IAM Role \(Tools for Windows PowerShell\) \(p. 212\)](#)
- AWS API : [Switching to an IAM Role \(API\) \(p. 214\)](#)

Overview

The following diagram illustrates the flow for SAML-enabled single sign-on.

Note

This specific use of SAML differs from the more general one illustrated at [About SAML 2.0-based Federation \(p. 149\)](#) because this workflow opens the AWS Management Console on behalf of the user. This requires the use of the AWS SSO endpoint instead of directly calling the `AssumeRoleWithSAML` API. The endpoint calls the API for the user and returns a URL that automatically redirects the user's browser to the AWS Management Console.



The diagram illustrates the following steps:

1. The user browses to your organization's portal and selects the option to go to the AWS Management Console. In your organization, the portal is typically a function of your identity provider (IdP) that handles the exchange of trust between your organization and AWS. For example, in Active Directory Federation Services, the portal URL is: `https://ADFSServiceName/adfs/ls/IdpInitiatedSignOn.aspx`
2. The portal verifies the user's identity in your organization.

3. The portal generates a SAML authentication response that includes assertions that identify the user and include attributes about the user. You can also configure your IdP to include a SAML assertion attribute called `SessionDuration` that specifies how long the console session is valid. The portal sends this response to the client browser.
4. The client browser is redirected to the AWS single sign-on endpoint and posts the SAML assertion.
5. The endpoint requests temporary security credentials on behalf of the user and creates a console sign-in URL that uses those credentials.
6. AWS sends the sign-in URL back to the client as a redirect.
7. The client browser is redirected to the AWS Management Console. If the SAML authentication response includes attributes that map to multiple IAM roles, the user is first prompted to select the role to use for access to the console.

From the user's perspective, the process happens transparently: The user starts at your organization's internal portal and ends up at the AWS Management Console, without ever having to supply any AWS credentials.

Consult the following sections for an overview of how to configure this behavior along with links to detailed steps.

Configure your network as a SAML provider for AWS

Inside your organization's network, you configure your identity store (such as Windows Active Directory) to work with a SAML-based identity provider (IdP) like Windows Active Directory Federation Services, Shibboleth, etc. Using your IdP, you generate a metadata document that describes your organization as an identity provider and includes authentication keys. You also configure your organization's portal to route user requests for the AWS Management Console to the AWS SAML endpoint for authentication using SAML assertions. How you configure your IdP to produce the `metadata.xml` file depends on your IdP. Refer to your IdP's documentation for instructions, or see [Integrating Third-Party SAML Solution Providers with AWS \(p. 161\)](#) for links to the web documentation for many of the SAML providers supported.

Create a SAML provider in IAM

Next, you sign in to the AWS Management Console and go to the IAM console. There you create a new SAML provider, which is an entity in IAM that holds information about your organization's identity provider. As part of this process, you upload the metadata document produced by the IdP software in your organization in the previous section. For details, see [Creating SAML Identity Providers \(p. 158\)](#).

Configure permissions in AWS for your federated users

The next step is to create an IAM role that establishes a trust relationship between IAM and your organization's IdP that identifies your IdP as a principal (trusted entity) for purposes of federation. The role also defines what users authenticated by your organization's IdP are allowed to do in AWS. You can use the IAM console to create this role. When you create the trust policy that indicates who can assume the role, you specify the SAML provider that you created earlier in IAM along with one or more SAML attributes that a user must match to be allowed to assume the role. For example, you can specify that only users whose SAML `eduPersonOrgDN` value is `ExampleOrg` are allowed to sign in. The role wizard automatically adds a condition to test the `saml:aud` attribute to make sure that the role is assumed only for sign-in to the AWS Management Console. The trust policy for the role might look like this:

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Principal": {"Federated": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:saml-provider/  
ExampleOrgSSOProvider"},  
        "Action": "sts:AssumeRoleWithSAML",  
        "Condition": {"StringEquals": {
```

```
        "saml:edupersonorgdn": "ExampleOrg",
        "saml:aud": "https://signin.aws.amazon.com/saml"
    }}
}]
```

For the [permission policy \(p. 275\)](#) in the role, you specify permissions as you would for any role, user, or group. For example, if users from your organization are allowed to administer Amazon EC2 instances, you explicitly allow Amazon EC2 actions in the permission policy. You can do this by assigning a [managed policy \(p. 331\)](#), such as the [Amazon EC2 Full Access](#) managed policy.

For details about creating a role for a SAML IdP, see [Creating a Role for SAML 2.0 Federation \(Console\) \(p. 200\)](#).

Finish configuring the SAML IdP and create assertions for the SAML authentication response

After you create the role, inform your SAML IdP about AWS as a service provider by installing the `saml-metadata.xml` file found at <https://signin.aws.amazon.com/static/saml-metadata.xml>. How you install that file depends on your IdP. Some providers give you the option to type the URL, whereupon the IdP gets and installs the file for you. Others require you to download the file from the URL and then provide it as a local file. Refer to your IdP documentation for details, or see [Integrating Third-Party SAML Solution Providers with AWS \(p. 161\)](#) for links to the web documentation for many of the supported SAML providers.

You also configure the information that you want the IdP to pass as SAML attributes to AWS as part of the authentication response. Most of this information appears in AWS as condition context keys that you can evaluate in your policies to ensure that only authorized users in the right contexts are granted permissions to access your AWS resources. You can specify time windows that restrict when the console may be used and the maximum time (up to 12 hours) that users can access the console before having to refresh their credentials. For details, see [Configuring SAML Assertions for the Authentication Response \(p. 163\)](#).

Creating a URL that Enables Federated Users to Access the AWS Management Console (Custom Federation Broker)

You can write and run code to create a URL that lets users who sign in to your organization's network securely access the AWS Management Console. The URL includes a sign-in token that you get from AWS and that authenticates the user to AWS.

Note

If your organization uses an identity provider (IdP) that is compatible with SAML, you can set up access to the console without writing code. This works with providers like Microsoft's Active Directory Federation Services or open-source Shibboleth. For details, see [Enabling SAML 2.0 Federated Users to Access the AWS Management Console \(p. 167\)](#).

To enable your organization's users to access the AWS Management Console, you can create a custom "identity broker" that performs the following steps:

1. Verify that the user is authenticated by your local identity system.
2. Call the AWS Security Token Service (AWS STS) [AssumeRole](#) (recommended) or [GetFederationToken](#) API operations to obtain temporary security credentials for the user. The credentials are associated with a role with permissions that control what the user can do. These credentials have a maximum duration as specified in the `DurationSeconds` parameter of the [AssumeRole](#) or [GetFederationToken](#) API call used to generate them.

Important

If you use the [AssumeRole](#) API, you must call it as an IAM user with long-term credentials. The call to the federation endpoint in step 3 works only if the temporary credentials are requested by an IAM user with long-term credentials. If the temporary credentials are

requested by an IAM assumed-role user with a different set of temporary credentials, then the call to the federation endpoint fails.

3. Call the AWS federation endpoint and supply the temporary security credentials to request a sign-in token.
 - If you used one of the `AssumeRole*` API operations to get the temporary security credentials, then this request can include the `SessionDuration` parameter. This parameter specifies how long the federated console session is valid, up to a maximum of 12 hours.
 - If you instead used the `GetFederationToken` API to get the credentials, then you don't need the `SessionDuration` parameter. The temporary credentials are already valid for up to 36 hours and specify the maximum length of the federated console session.
4. Construct a URL for the console that includes the token.
5. Give the URL to the user or invoke the URL on the user's behalf.

The URL that the federation endpoint provides is valid for 15 minutes after it is created. The temporary security credentials associated with the URL are valid for the duration you specified when you created them, starting from the time they were created.

Important

The URL grants access to your AWS resources through the AWS Management Console if you have enabled permissions in the associated temporary security credentials. For this reason, you should treat the URL as a secret. We recommend returning the URL through a secure redirect, for example, by using a 302 HTTP response status code over an SSL connection. For more information about the 302 HTTP response status code, go to [RFC 2616, section 10.3.3](#).

To view a sample application that shows you how you can implement a single sign-on solution, go to [AWS Management Console federation proxy sample use case](#) in the *AWS Sample Code & Libraries*.

To complete these tasks, you can use the [HTTPS Query API for AWS Identity and Access Management \(IAM\)](#) and the [AWS Security Token Service \(AWS STS\)](#). Or, you can use programming languages, such as Java, Ruby, or C#, along with the appropriate [AWS SDK](#). Each of these methods is described in the following sections.

Topics

- [Example Code Using IAM Query API Operations \(p. 170\)](#)
- [Example Code Using Python \(p. 173\)](#)
- [Example Code Using Java \(p. 174\)](#)
- [Example Showing How to Construct the URL \(Ruby\) \(p. 175\)](#)

Example Code Using IAM Query API Operations

You can construct a URL that gives your federated users direct access to the AWS Management Console. This task uses the IAM and AWS STS HTTPS Query API. For more information about making query requests, see [Making Query Requests](#).

Note

The following procedure contains examples of text strings. To enhance readability, line breaks have been added to some of the longer examples. When you create these strings for your own use, you should omit any line breaks.

To give a federated user access to your resources from the AWS Management Console

1. Authenticate the user in your identity and authorization system.
2. Obtain temporary security credentials for the user. The temporary credentials consist of an access key ID, a secret access key, and a security token. For more information about creating temporary credentials, see [Temporary Security Credentials \(p. 231\)](#).

To get temporary credentials, you call either the AWS STS [AssumeRole](#) API (recommended) or the [GetFederationToken](#) API. For more information about the differences between these API operations, see [Understanding the API Options for Securely Delegating Access to Your AWS Account](#) in the AWS Security Blog.

Important

- When you create temporary security credentials, you must specify the permissions the credentials grants to the user who holds them. For any of the API operations that begin with `AssumeRole*`, you use an IAM role to assign permissions. For the other API operations, the mechanism varies with the API. For more details, see [Controlling Permissions for Temporary Security Credentials \(p. 246\)](#).
 - If you use the `AssumeRole` API, you must call it as an IAM user with long-term credentials. The call to the federation endpoint in step 3 works only if the temporary credentials are requested by an IAM user with long-term credentials. If the temporary credentials are requested by an IAM assumed-role user with a different set of temporary credentials, then the call to the federation endpoint fails.
3. After you obtain the temporary security credentials, build them into a JSON "session" string to exchange them for a sign-in token. The following example shows how to encode the credentials. You replace the placeholder text with the appropriate values from the credentials that you receive in the previous step.

```
{"sessionId": "*** temporary access key ID ***",
 "sessionKey": "*** temporary secret access key ***",
 "sessionToken": "*** security token ***"}
```

4. [URL encode](#) the session string from the previous step. Because the information that you are encoding is sensitive, we recommend that you avoid using a web service for this encoding. Instead, use a locally installed function or feature in your development toolkit to securely encode this information. You can use the `urllib.quote_plus` function in Python, the `URLEncoder.encode` function in Java, or the `CGI.escape` function in Ruby. See the examples later in this topic.
5. Send your request to the AWS federation endpoint at the following address:

<https://signin.aws.amazon.com/federation>

The request must include the `Action` and `Session` parameters, and (optionally) if you used `AssumeRole`, a `SessionDuration` HTTP parameter as shown in the following example.

```
Action = getSigninToken
SessionDuration = time in seconds
Session = *** the URL encoded JSON string created in steps 3 & 4 ***
```

The `SessionDuration` parameter specifies the number of seconds that the credentials for the console session are valid. This is separate from the duration of the temporary credentials. You can specify a `SessionDuration` maximum value of 43200 (12 hours). If the parameter is missing, then the session defaults to the duration of the credentials you retrieved from AWS STS in step 2 (which defaults to one hour). See the [documentation for the AssumeRole API](#) for details about how to specify duration using the `DurationSeconds` parameter. The ability to create a console session that is longer than one hour is intrinsic to the `getSigninToken` action of the federation endpoint. You cannot use the IAM or STS API operations to get credentials that are valid for longer than one hour (3600 seconds).

Note

You do not need the `SessionDuration` HTTP parameter if you got the temporary credentials with `GetFederationToken`. The console session can be as long as the temporary credentials are valid (up to 36 hours).

When you enable console sessions with an extended duration the risk of compromise of the credentials rises. To help you mitigate this risk, you can immediately disable the active console sessions for any role by choosing **Revoke Sessions** on the **Role Summary** IAM console page. For more information, see [Revoking IAM Role Temporary Security Credentials \(p. 220\)](#).

The following is an example of what your request might look like. The lines are wrapped here for readability, but you should submit it as a one-line string.

```
https://signin.aws.amazon.com/federation
?Action=getSignInToken
&SessionDuration=43200
&Session=%7B%22sessionId%22%3A+%22ASIAJUMHIZPTOKTBMK5A%22%2C+%22sessionKey%22
%3A+%22LSD7LWI%2FL%2FN%2BgYpan5QFz0XUpc8s7HYjRsgcsrsrn%22%2C+%22sessionToken%22
%3A+%22FQoDYXzdEBQaDlbj3VWv2u50NN%2F3yyLSASwYtWhPnGPMNmzZFFzsL0Qd3vtYHw5A5dW
AjOsrkPkgHomie3mJip5%2F0djDBbo7Sm0%2FENDEiCdp$QKodTpleKA8xQq0CwFg6a69xdEBQT8
FipATnLbKoyS4b%2FebhnsTUjZZQWp0wXXqFF7gSm%2FMe2tXe0jzsdP001obeZ91ijPSdF1k2b5
PfGhiuyAR9ad5%2BbM0pY86fKex1qsyjtjvyTbZ9nXe6DvxVDcnCohOGETJ7XFkSFdH0v%2FYR25C
UAhJ3nXIkIbG7Ucv9cOEpCf%2Fg23ijRgILIBQ%3D%3D%22%7D
```

The response from the federation endpoint is a JSON document with an `SignInToken` value. It will look similar to the following example.

```
{"SignInToken": "*** the SignInToken string ***"}
```

- Finally, create the URL that your federated users can use to access the AWS Management Console. The URL is the same federation URL endpoint that you used in [Step 5 \(p. 171\)](#), plus the following parameters:

```
?Action = login
&Issuer = *** the form-urlencoded URL for your internal sign-in page ***
&Destination = *** the form-urlencoded URL to the desired AWS console page ***
&SignInToken = *** the value of SignInToken received in the previous step ***
```

The following example shows what the final URL might look like. The URL is valid for 15 minutes from the time it is created. The temporary security credentials and console session embedded within the URL are valid for the duration you specify in the `SessionDuration` parameter when you initially request them.

```
https://signin.aws.amazon.com/federation
?Action=login
&Issuer=https%3A%2F%2Fexample.com
&Destination=https%3A%2F%2Fconsole.aws.amazon.com%2Fs
&SignInToken=VCQgs5qZzT3Q6fn8Tr5EXAMPLELnwB7JjUC-SHwnUUWabcRdnWsi4DBn-dvC
CZ85wrD0nmlUcZEXAMPLE-vXYH4Q__mleuF_W2BE5H4exbe9y40f-kje53SsjNNecATfjIzpW1
WibbnH6YcYRiBoffZBGExbEXAMPLE5aiKX4THWjQKC6gg6alHu6JFrnOJoK3dtP6I9a6hi6yPgm
iOkPZMmNGmhsVxetKzr8mx3pxhHbMEXAMPLEtv1pij0rok3IyCR2YVcIjqwfWv32HU2Xljq47iu
3fU6uOfUComeKiqTGX974xzJOZbdmX_t_lLrhEXAMPLEDDIisSnyHgw2xaZzqudm4mo2uTDk9Pv
915K0ZCqIgEXAMPLEcA6tgLPykEWGUyH6BdSC6166n4M4JkXI0gac7_7821YqixsNxZ6rsrpzwf
nQoS1407R0eJCCJ684EXAMPLERdBNuLbUYpz2Iw3vIN0tQgOujwnwydPscM9F7foaEK3jwMkg
Apeb1-6L_OB12MZhufxx5555EXAMPLEhyETEd4ZulKPdXHkg16t9zklHz2Uy1RUTUhUxNtSQ
nWc5xkbBoEcXqposIeK7yhje9Vzhed1AEXAMPLElbWeouACEMG6-Vd3dAgFYd6i5FYoyFrZLWvm
0LSG7RyYKeYN5ViZUk3YWQpyjPORiT5KUrsUi-NEXAMPLExMOMdoODBeGKQsk-1u2ozh6r8bxwCRNhujg
```

Example Code Using Python

The following example shows how to use Python to programmatically construct a URL that gives federated users direct access to the AWS Management Console. The example uses the [AWS SDK for Python \(Boto\)](#).

The code uses the [AssumeRole](#) API to obtain temporary security credentials.

```
import urllib, json
import requests # 'pip install requests'
from boto.sts import STSConnection # AWS SDK for Python (Boto) 'pip install boto'

# Step 1: Authenticate user in your own identity system.

# Step 2: Using the access keys for an IAM user in your AWS account,
# call "AssumeRole" to get temporary access keys for the federated user

# Note: Calls to AWS STS AssumeRole must be signed using the access key ID
# and secret access key of an IAM user or using existing temporary credentials.
# The credentials can be in EC2 instance metadata, in environment variables,
# or in a configuration file, and will be discovered automatically by the
# STSConnection() function. For more information, see the Python SDK docs:
# http://boto.readthedocs.org/en/latest/boto_config_tut.html
sts_connection = STSConnection()

assumed_role_object = sts_connection.assume_role(
    role_arn="arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:role/ROLE-NAME",
    role_session_name="AssumeRoleSession"
)

# Step 3: Format resulting temporary credentials into JSON
json_string_with_temp_credentials = '{'
json_string_with_temp_credentials += '"sessionId":' + \
    assumed_role_object.credentials.access_key + '",'
json_string_with_temp_credentials += '"sessionKey":' + \
    assumed_role_object.credentials.secret_key + '",'
json_string_with_temp_credentials += '"sessionToken":' + \
    assumed_role_object.credentials.session_token + "'"
json_string_with_temp_credentials += '}'"

# Step 4. Make request to AWS federation endpoint to get sign-in token. Construct the
# parameter string with
# the sign-in action request, a 12-hour session duration, and the JSON document with
# temporary credentials
# as parameters.
request_parameters = "?Action=getSigninToken"
request_parameters += "&SessionDuration=43200"
request_parameters += "&Session=" + urllib.quote_plus(json_string_with_temp_credentials)
request_url = "https://signin.aws.amazon.com/federation" + request_parameters
r = requests.get(request_url)
# Returns a JSON document with a single element named SigninToken.
signin_token = json.loads(r.text)

# Step 5: Create URL where users can use the sign-in token to sign in to
# the console. This URL must be used within 15 minutes after the
# sign-in token was issued.
request_parameters = "?Action=login"
request_parameters += "&Issuer=Example.org"
request_parameters += "&Destination=" + urllib.quote_plus("https://"
console.aws.amazon.com/")
request_parameters += "&SigninToken=" + signin_token["SigninToken"]
request_url = "https://signin.aws.amazon.com/federation" + request_parameters

# Send final URL to stdout
```

```
print request_url
```

Example Code Using Java

The following example shows how to use Java to programmatically construct a URL that gives federated users direct access to the AWS Management Console. The following code snippet uses the [AWS SDK for Java](#).

```
import java.net.URLEncoder;
import java.net.URL;
import java.netURLConnection;
import java.io.BufferedReader;
import java.io.InputStreamReader;
// Available at http://www.json.org/java/index.html
import org.json.JSONObject;
import com.amazonaws.auth.AWS Credentials;
import com.amazonaws.auth.BasicAWS Credentials;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient;
import com.amazonaws.services.securitytoken.model.Credentials;
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;

/* Calls to AWS STS APIs must be signed using the access key ID
   and secret access key of an IAM user or using existing temporary
   credentials. The credentials should not be embedded in code. For
   this example, the code looks for the credentials in a
   standard configuration file.
*/
AWS Credentials credentials =
    new PropertiesCredentials(
        AwsConsoleApp.class.getResourceAsStream("AwsCredentials.properties"));

AWSSecurityTokenServiceClient stsClient =
    new AWSSecurityTokenServiceClient(credentials);

GetFederationTokenRequest getFederationTokenRequest =
    new GetFederationTokenRequest();
getFederationTokenRequest.setDurationSeconds(3600);
getFederationTokenRequest.setName("UserName");

// A sample policy for accessing Amazon Simple Notification Service (Amazon SNS) in the
// console.

String policy = "{\"Version\":\"2012-10-17\", \"Statement\":[{\"Action\":\"sns:*\", \" +
    "\"Effect\":\"Allow\", \"Resource\":\"*\"]}}";
getFederationTokenRequest.setPolicy(policy);

GetFederationTokenResult federationTokenResult =
    stsClient.getFederationToken(getFederationTokenRequest);

Credentials federatedCredentials = federationTokenResult.getCredentials();

// The issuer parameter specifies your internal sign-in
// page, for example https://mysignin.internal.mycompany.com/.
// The console parameter specifies the URL to the destination console of the
// AWS Management Console. This example goes to Amazon SNS.
// The signin parameter is the URL to send the request to.

String issuerURL = "https://mysignin.internal.mycompany.com/";
String consoleURL = "https://console.aws.amazon.com/sns";
String signInURL = "https://signin.aws.amazon.com/federation";
```

```

// Create the sign-in token using temporary credentials,
// including the access key ID, secret access key, and security token.
String sessionJson = String.format(
    "{\"%1$s\":\"%2$s\", \"%3$s\": \"%4$s\", \"%5$s\": \"%6$s\"}",
    "sessionId", federatedCredentials.getAccessKeyId(),
    "sessionKey", federatedCredentials.getSecretAccessKey(),
    "sessionToken", federatedCredentials.getSessionToken());

// Construct the sign-in request with the request sign-in token action, a
// 12-hour console session duration, and the JSON document with temporary
// credentials as parameters.

String getSigninTokenURL = signInURL +
    "?Action=getSigninToken" +
    "&SessionDuration=43200" +
    "&SessionType=json&Session=" +
    URLEncoder.encode(sessionJson, "UTF-8");

URL url = new URL(getSigninTokenURL);

// Send the request to the AWS federation endpoint to get the sign-in token
URLConnection conn = url.openConnection();

BufferedReader bufferReader = new BufferedReader(new
    InputStreamReader(conn.getInputStream()));
String returnContent = bufferReader.readLine();

String signinToken = new JSONObject(returnContent).getString("SigninToken");

String signinTokenParameter = "&SigninToken=" + URLEncoder.encode(signinToken, "UTF-8");

// The issuer parameter is optional, but recommended. Use it to direct users
// to your sign-in page when their session expires.

String issuerParameter = "&Issuer=" + URLEncoder.encode(issuerURL, "UTF-8");

// Finally, present the completed URL for the AWS console session to the user

String destinationParameter = "&Destination=" + URLEncoder.encode(consoleURL, "UTF-8");
String loginURL = signInURL + "?Action=login" +
    signinTokenParameter + issuerParameter + destinationParameter;

```

Example Showing How to Construct the URL (Ruby)

The following example shows how to use Ruby to programmatically construct a URL that gives federated users direct access to the AWS Management Console. This code snippet uses the [AWS SDK for Ruby](#).

```

require 'rubygems'
require 'json'
require 'open-uri'
require 'cgi'
require 'aws-sdk'

# Create a new STS instance
#
# Note: Calls to AWS STS APIs must be signed using an access key ID
# and secret access key. The credentials can be in EC2 instance metadata
# or in environment variables and will be automatically discovered by
# the default credentials provider in the AWS Ruby SDK.
sts = Aws::STS::Client.new()

# The following call creates a temporary session that returns
# temporary security credentials and a session token.
# The policy grants permissions to work

```

```

# in the AWS SNS console.

session = sts.get_federation_token({
    duration_seconds: 3600,
    name: "UserName",
    policy: "{\"Version\":\"2012-10-17\", \"Statement\":{\"Effect\":\"Allow\", \"Action\": \"sns:*\", \"Resource\":\"*\"}}",
})

# The issuer value is the URL where users are directed (such as
# to your internal sign-in page) when their session expires.
#
# The console value specifies the URL to the destination console.
# This example goes to the Amazon SNS console.
#
# The sign-in value is the URL of the AWS STS federation endpoint.
issuer_url = "https://mysignin.internal.mycompany.com/"
console_url = "https://console.aws.amazon.com/sns"
signin_url = "https://signin.aws.amazon.com/federation"

# Create a block of JSON that contains the temporary credentials
# (including the access key ID, secret access key, and session token).
session_json = {
    :sessionId => session.credentials[:access_key_id],
    :sessionKey => session.credentials[:secret_access_key],
    :sessionToken => session.credentials[:session_token]
}.to_json

# Call the federation endpoint, passing the parameters
# created earlier and the session information as a JSON block.
# The request returns a sign-in token that's valid for 15 minutes.
# Signing in to the console with the token creates a session
# that is valid for 12 hours.
get_signin_token_url = signin_url +
    "?Action=getSignInToken" +
    "&SessionType=json&Session=" +
    CGI.escape(session_json)

returned_content = URI.parse(get_signin_token_url).read

# Extract the sign-in token from the information returned
# by the federation endpoint.
signin_token = JSON.parse(returned_content)['SigninToken']
signin_token_param = "&SigninToken=" + CGI.escape(signin_token)

# Create the URL to give to the user, which includes the
# sign-in token and the URL of the console to open.
# The "issuer" parameter is optional but recommended.
issuer_param = "&Issuer=" + CGI.escape(issuer_url)
destination_param = "&Destination=" + CGI.escape(console_url)
login_url = signin_url + "?Action=login" + signin_token_param +
    issuer_param + destination_param

```

Using Service-Linked Roles

A service-linked role is a unique type of IAM role that is linked directly to an AWS service. Service-linked roles are predefined by the service and include all the permissions that the service requires to call other AWS services on your behalf. The linked service also defines how you create, modify, and delete a service-linked role. A service might automatically create or delete the role. It might allow you to create, modify, or delete the role as part of a wizard or process in the service. Or it might require that you use IAM to create or delete the role. Regardless of the method, service-linked roles make setting up a service easier because you don't have to manually add the necessary permissions.

The linked service defines the permissions of its service-linked roles, and unless defined otherwise, only that service can assume the roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete the roles only after first deleting their related resources. This protects your resources because you can't inadvertently remove permission to access the resources.

For information about which services support using service-linked roles, see [AWS Services That Work with IAM \(p. 417\)](#) and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-Linked Role Permissions

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create or edit the description of a service-linked role.

To allow an IAM entity to create a specific service-linked role

Add the following policy to the IAM entity that needs to create the service-linked role.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX*",
            "Condition": {"StringLike": {"iam:AWSServiceName": "SERVICE-NAME.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX*"
        }
    ]
}
```

To allow an IAM entity to create any service-linked role

Add the following statement to the permissions policy for the IAM entity that needs to create a service-linked role, or any service role that includes the needed policies. This policy statement does not allow the IAM entity to attach a policy to the role.

```
{
    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

To allow an IAM entity to edit the description of any service roles

Add the following statement to the permissions policy for the IAM entity that needs to edit the description of a service-linked role, or any service role.

```
{
    "Effect": "Allow",
    "Action": "iam:UpdateRoleDescription",
```

```
    "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

To allow an IAM entity to delete a specific service-linked role

Add the following statement to the permissions policy for the IAM entity that needs to delete the service-linked role.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX*"
}
```

To allow an IAM entity to delete any service role

Add the following statement to the permissions policy for the IAM entity that needs to delete a service-linked role, or any service-role.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

Alternatively, you can use an AWS managed policy to provide full access to the service.

Creating a Service-Linked Role

The method that you use to create a service-linked role depends on the service. In some cases, you don't need to manually create a service-linked role. For example, when you complete a specific action (such as creating a resource) in the service, the service might create the service-linked role for you. Or if you were using a service before it began supporting service-linked roles, then the service might have automatically created the role in your account. To learn more, see [A New Role Appeared in My AWS Account \(p. 400\)](#).

In other cases, the service might support creating a service-linked role manually using the service console, API, or CLI. For information about which services support using service-linked roles, see [AWS Services That Work with IAM \(p. 417\)](#) and look for the services that have Yes in the **Service-Linked Role** column. To learn whether the service supports creating the service-linked role, choose the Yes link to view the service-linked role documentation for that service.

If the service does not support creating the role, then you can use IAM to create the service-linked role.

Important

Service-linked roles count toward your [IAM roles in an AWS account limit](#), but if you have reached your limit, you can still create service-linked roles in your account. Only service-linked roles can exceed the limit.

Creating a Service-Linked Role (IAM Console)

Before you create a service-linked role in IAM, find out whether the linked service automatically creates service-linked roles and whether you can't create the role from the service's console, API, or CLI.

To create a service-linked role (console)

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 of the IAM console, choose **Roles**. Then choose **Create role**.
3. Choose the **AWS Service** role type, and then choose the service that you want to allow to assume this role.
4. Choose the use case for your service. If the specified service has only one use case, it is selected for you. Use cases are defined by the service to include the trust policy required by the service. Then choose **Next: Permissions**.
5. Choose one or more permissions policies to attach to the role. Depending on the use case that you selected, the service might do any of the following:
 - Define the permissions used by the role
 - Allow you to choose from a limited set of permissions
 - Allow you to choose from any permissions
 - Allow you to select no policies at this time, create the policies later, and then attach them to the role.

Select the box next to the policy that assigns the permissions that you want the role to have, and then choose **Next: Review**.

Note

The permissions that you specify are available to any entity that uses the role. By default, a role has no permissions.

6. For **Role name**, the degree of role name customization is defined by the service. If the service defines the role's name, then this option is not editable. In other cases, the service might define a prefix for the role and allow you to type an optional suffix.

If possible, type a role name suffix to add to the default name. This suffix helps you identify the purpose of this role. Role names must be unique within your AWS account. They are not distinguished by case. For example, you cannot create roles named both `<service-linked-role-name>_SAMPLE` and `<service-linked-role-name>_sample`. Because various entities might reference the role, you cannot edit the name of the role after it has been created.

7. (Optional) For **Role description**, edit the description for the new service-linked role.
8. Review the role and then choose **Create role**.

Creating a Service-Linked Role (IAM CLI)

Before creating a service-linked role in IAM, find out whether the linked service automatically creates service-linked roles and whether you can create the role from the service's CLI. If the service CLI is not supported, you can use IAM commands to create a service-linked role with the trust policy and inline policies that the service needs to assume the role.

To create a service-linked role (IAM CLI)

Use the following command:

```
$ aws iam create-service-linked-role --aws-service-name SERVICE-NAME.amazonaws.com
```

Creating a Service-Linked Role (IAM API)

Before creating a service-linked role in IAM, find out whether the linked service automatically creates service-linked roles and whether you can create the role from the service's API. If the service API is not

supported, you can use the IAM API to create a service-linked role with the trust policy and inline policies that the service needs to assume the role.

To create a service-linked role (API)

Use the [CreateServiceLinkedRole](#) API call. In the request, specify a service name of *SERVICE_NAME_URL*.amazonaws.com.

For example, to create the **Lex Bots** service-linked role, use lex.amazonaws.com.

Editing a Service-Linked Role

The method that you use to edit a service-linked role depends on the service. Some services might allow you to edit the permissions for a service-linked role from the service console, API, or CLI. However, after you create a service-linked role, you cannot change the name of the role because various entities might reference the role. You can edit the description of any role from the IAM console, API, or CLI.

For information about which services support using service-linked roles, see [AWS Services That Work with IAM \(p. 417\)](#) and look for the services that have **Yes** in the **Service-Linked Role** column. To learn whether the service supports editing the service-linked role, choose the **Yes** link to view the service-linked role documentation for that service.

Editing a Service-Linked Role Description (IAM Console)

You can use the IAM console to edit the description of a service-linked role.

To edit the description of a service-linked role (console)

1. In the navigation pane of the IAM console, choose **Roles**.
2. Choose the name of the role to modify.
3. To the far right of **Role description**, choose **Edit**.
4. Type a new description in the box and choose **Save**.

Editing a Service-Linked Role Description (IAM CLI)

You can use IAM commands from the AWS Command Line Interface to edit the description of a service-linked role.

To change the description of a service-linked role (IAM CLI)

1. (Optional) To view the current description for a role, use the following command:

```
$ aws iam get-role --role-name ROLE-NAME
```

Use the role name, not the ARN, to refer to roles with the CLI commands. For example, if a role has the following ARN: arn:aws:iam::123456789012:role/myrole, you refer to the role as **myrole**.

2. To update a service-linked role's description, use the following command:

```
$ aws iam update-role-description --role-name ROLE-NAME --description OPTIONAL-DESCRIPTION
```

Editing a Service-Linked Role Description (IAM API)

You can use the IAM API to edit the description of a service-linked role.

To change the description of a service-linked role (API)

1. (Optional) To view the current description for a role, use the following command, and specify the name of the role:

IAM API: [GetRole](#)

2. To update a role's description, use the following command, and specify the name (and optional description) of the role:

IAM API: [UpdateRoleDescription](#)

Deleting a Service-Linked Role

The method that you use to create a service-linked role depends on the service. In some cases, you don't need to manually delete a service-linked role. For example, when you complete a specific action (such as removing a resource) in the service, the service might delete the service-linked role for you.

In other cases, the service might support deleting a service-linked role manually from the service console, API, or CLI.

For information about which services support using service-linked roles, see [AWS Services That Work with IAM \(p. 417\)](#) and look for the services that have **Yes** in the **Service-Linked Role** column. To learn whether the service supports deleting the service-linked role, choose the **Yes** link to view the service-linked role documentation for that service.

If the service does not support deleting the role, then you can delete the service-linked role from the IAM console, API, or CLI. If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up your service-linked role before you can delete it.

Cleaning Up a Service-Linked Role

Before you can use IAM to delete a service-linked role, you must first confirm that the role has no active sessions and remove any resources used by the role.

To check whether the service-linked role has an active session in the IAM console

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 of the IAM console, choose **Roles**. Then choose the name (not the check box) of the service-linked role.
3. On the **Summary** page for the selected role, choose the **Access Advisor** tab.
4. On the **Access Advisor** tab, review recent activity for the service-linked role.

Note

If you are unsure whether the service is using the service-linked role, you can try to delete the role. If the service is using the role, then the deletion fails and you can view the regions where the role is being used. If the role is being used, then you must wait for the session to end before you can delete the role. You cannot revoke the session for a service-linked role.

To remove resources used by a service-linked role

For information about which services support using service-linked roles, see [AWS Services That Work with IAM \(p. 417\)](#) and look for the services that have **Yes** in the **Service-Linked Role** column. To learn whether the service supports deleting the service-linked role, choose the **Yes** link to view the service-

linked role documentation for that service. See the documentation for that service to learn how to remove resources used by your service-linked role.

Deleting a Service-Linked Role (IAM Console)

You can use the IAM console to delete a service-linked role.

To delete a service-linked role (console)

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 of the IAM console, choose **Roles**. Then select the check box next to the role name that you want to delete, not the name or row itself.
3. For **Role actions** at the top of the page, choose **Delete role**.
4. In the confirmation dialog box, review the service last accessed data, which shows when each of the selected roles last accessed an AWS service. This helps you to confirm whether the role is currently active. If you want to proceed, choose **Yes, Delete** to submit the service-linked role for deletion.
5. Watch the IAM console notifications to monitor the progress of the service-linked role deletion. Because the IAM service-linked role deletion is asynchronous, after you submit the role for deletion, the deletion task can succeed or fail.
 - If the task succeeds, then the role is removed from the list and a notification of success appears at the top of the page.
 - If the task fails, you can choose **View details** or **View Resources** from the notifications to learn why the deletion failed. If the deletion fails because the role is using the service's resources, then the notification includes a list of resources, if the service returns that information. You can then [clean up the resources \(p. 181\)](#) and submit the deletion again.

Note

You might have to repeat this process several times, depending on the information that the service returns. For example, your service-linked role might use six resources and your service might return information about five of them. If you clean up the five resources and submit the role for deletion again, the deletion fails and the service reports the one remaining resource. A service might return all of the resources, a few of them, or it might not report any resources.

- If the task fails and the notification does not include a list of resources, then the service might not return that information. To learn how to clean up the resources for that service, see [AWS Services That Work with IAM \(p. 417\)](#). Find your service in the table, and choose the **Yes** link to view the service-linked role documentation for that service.

Deleting a Service-Linked Role (IAM CLI)

You can use IAM commands from the AWS Command Line Interface to delete a service-linked role.

To delete a service-linked role (IAM CLI)

1. If you don't know the name of the service-linked role that you want to delete, type the following command to list the roles and their Amazon Resource Names (ARNs) in your account:

```
$ aws iam get-role --role-name role-name
```

Use the role name, not the ARN, to refer to roles with the CLI commands. For example, if a role has the following ARN: `arn:aws:iam::123456789012:role/myrole`, you refer to the role as `myrole`.

2. Because a service-linked role cannot be deleted if it is being used or has associated resources, you must submit a deletion request. That request can be denied if these conditions are not met. You

must capture the `deletion-task-id` from the response to check the status of the deletion task. Type the following command to submit a service-linked role deletion request:

```
$ aws iam delete-service-linked-role --role-name role-name
```

3. Type the following command to check the status of the deletion task:

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

The status of the deletion task can be NOT_STARTED, IN_PROGRESS, SUCCEEDED, or FAILED. If the deletion fails, the call returns the reason that it failed so that you can troubleshoot. If the deletion fails because the role is using the service's resources, then the notification includes a list of resources, if the service returns that information. You can then [clean up the resources \(p. 181\)](#) and submit the deletion again.

Note

You might have to repeat this process several times, depending on the information that the service returns. For example, your service-linked role might use six resources and your service might return information about five of them. If you clean up the five resources and submit the role for deletion again, the deletion fails and the service reports the one remaining resource. A service might return all of the resources, a few of them, or it might not report any resources. To learn how to clean up the resources for a service that does not report any resources, see [AWS Services That Work with IAM \(p. 417\)](#). Find your service in the table, and choose the Yes link to view the service-linked role documentation for that service.

Deleting a Service-Linked Role (IAM API)

You can use the IAM API to delete a service-linked role.

To delete a service-linked role (API)

1. To submit a deletion request for a service-linked role, call [DeleteServiceLinkedRole](#). In the request, specify a role name.

Because a service-linked role cannot be deleted if it is being used or has associated resources, you must submit a deletion request. That request can be denied if these conditions are not met. You must capture the `DeletionTaskId` from the response to check the status of the deletion task.

2. To check the status of the deletion, call [GetServiceLinkedRoleDeletionStatus](#). In the request, specify the `DeletionTaskId`.

The status of the deletion task can be NOT_STARTED, IN_PROGRESS, SUCCEEDED, or FAILED. If the deletion fails, the call returns the reason that it failed so that you can troubleshoot. If the deletion fails because the role is using the service's resources, then the notification includes a list of resources, if the service returns that information. You can then [clean up the resources \(p. 181\)](#) and submit the deletion again.

Note

You might have to repeat this process several times, depending on the information that the service returns. For example, your service-linked role might use six resources and your service might return information about five of them. If you clean up the five resources and submit the role for deletion again, the deletion fails and the service reports the one remaining resource. A service might return all of the resources, a few of them, or it might not report any resources. To learn how to clean up the resources for a service that does not report any resources, see [AWS Services That Work with IAM \(p. 417\)](#). Find your service in the table, and choose the Yes link to view the service-linked role documentation for that service.

Creating IAM Roles

To create a role, you can use the AWS Management Console, the AWS CLI, the Tools for Windows PowerShell, or the IAM API.

If you use the AWS Management Console, a wizard guides you through the steps for creating a role. The wizard has slightly different steps depending on whether you're creating a role for an AWS service, for an AWS account, or for a federated user.

Topics

- [Creating a Role to Delegate Permissions to an IAM User \(p. 184\)](#)
- [Creating a Role to Delegate Permissions to an AWS Service \(p. 191\)](#)
- [Creating a Role for a Third-Party Identity Provider \(Federation\) \(p. 194\)](#)
- [Examples of Policies for Delegating Access \(p. 201\)](#)

Creating a Role to Delegate Permissions to an IAM User

You can use IAM roles to delegate access to your AWS resources. With IAM roles, you can establish trust relationships between your *trusting* account and other AWS *trusted* accounts. The trusting account owns the resource to be accessed and the trusted account contains the users who need access to the resource.

After you create the trust relationship, an IAM user or an application from the trusted account can use the AWS Security Token Service (AWS STS) `AssumeRole` API action. This action provides temporary security credentials that enable access to AWS resources in your account.

The accounts can both be controlled by you, or the account with the users can be controlled by a third party. If the other account with the users is in an AWS account that you do not control, then you can use the `externalID` attribute and a unique identifier supplied by the third-party account. This helps ensure that access occurs only in the correct contexts. For more information, see [How to Use an External ID When Granting Access to Your AWS Resources to a Third Party \(p. 187\)](#).

For information about how to use roles to delegate permissions, see [Roles Terms and Concepts \(p. 135\)](#). For information about using a service role to allow services to access resources in your account, see [Creating a Role to Delegate Permissions to an AWS Service \(p. 191\)](#).

Creating an IAM Role (Console)

You can use the AWS Management Console to create a role that an IAM user can assume. For example, imagine that your organization has multiple AWS accounts to isolate a development environment from a production environment. To view a high-level description of the steps necessary to set up and use a role that allows users in the development account to access resources in the production account, see [Example Scenario Using Separate Development and Production Accounts \(p. 139\)](#).

To create a role (console)

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 of the console, choose **Roles** and then choose **Create role**.
3. Choose the **Another AWS account** role type.
4. For **Account ID**, type the AWS account ID to which you want to grant access to your resources.

The administrator of the specified account can grant permission to assume this role to any IAM user in that account. To do this, the administrator attaches a policy to the user or a group that grants permission for the `sts:AssumeRole` action. That policy must specify the role's ARN as the **Resource**.

5. If you are granting permissions to users from an account that you do not control, select **Require external ID**. The external ID can be any word or number that is agreed upon between you and the administrator of the third party account. This option automatically adds a condition to the trust policy that allows the user to assume the role only if the request includes the correct `sts:ExternalID`. For more information, see [How to Use an External ID When Granting Access to Your AWS Resources to a Third Party \(p. 187\)](#).

Important

Choosing this option restricts access to the role only through the AWS CLI, Tools for Windows PowerShell, or the AWS API. This is because you cannot use the AWS console to switch to a role that has an `ExternalID` condition in its trust policy. However, you can create this kind of access programmatically by writing a script or an application using the relevant SDK. For more information and a sample script, see [How to Enable Cross-Account Access to the AWS Management Console](#) in the [AWS Security Blog](#).

6. If you want to restrict the role to users who sign in with multi-factor authentication (MFA), select **Require MFA**. This adds a condition to the role's trust policy that checks for an MFA sign-in. A user who wants to assume the role must sign in with a temporary one-time password from a configured MFA device. Users without MFA authentication cannot assume the role. For more information about MFA, see [Using Multi-Factor Authentication \(MFA\) in AWS \(p. 91\)](#)
7. Choose **Next: Permissions**.
8. Choose one or more permissions policies to attach to the role. Choose policies that specify what actions can be done on specific resources (similar to setting permissions for IAM groups). For information about using policies to manage permissions, see [IAM Policies \(p. 275\)](#).

The permissions that you specify are available to any entity that uses the role. By default, a role has no permissions.

Select the box next to the policy that assigns the permissions that you want the users to have. Then choose **Next: Review**. If you prefer, you can select no policies at this time, create the policies later, and then attach them to the role.

9. For **Role name**, type a name for your role. This name helps you identify the purpose of this role. Role names must be unique within your AWS account. They are not distinguished by case. For example, you cannot create roles named both `PRODROLE` and `prodrole`. Because various entities might reference the role, you cannot edit the name of the role after it has been created.
10. (Optional) For **Role description**, type a description for the new role.
11. Review the role and then choose **Create role**.

Important

Remember that this is only the first half of the configuration required. You must also give individual users in the trusted account permissions to switch to the role. For more information about this step, see [Granting a User Permissions to Switch Roles \(p. 205\)](#).

After you create the role and grant it permissions to perform AWS tasks or access AWS resources, the user can assume the role. For more information, see [Switching to a Role \(AWS Management Console\) \(p. 209\)](#).

Creating an IAM Role (AWS CLI)

Creating a role from the AWS CLI involves multiple steps. When you use the console to create a role, many of the steps are done for you, but with the AWS CLI you must explicitly perform each step yourself. You must create the policy and assign a permissions policy to the role.

To create a role for cross-account access (AWS CLI)

- Create a role: `aws iam create-role`
- Attach a managed permission policy to the role: `aws iam attach-role-policy`

-or-

Create an inline permission policy for the role: [aws iam put-role-policy](#)

The following example shows both steps in a simple environment. The example assumes that you are using a client computer running Windows, and have already configured your command line interface with your account credentials and region. For more information, see [Configuring the AWS Command Line Interface](#).

The sample trust policy referenced in the first command contains the following JSON code. It enables users in the account 123456789012 to assume the role, but only if the user provides MFA authentication. For more information about MFA, see [Using Multi-Factor Authentication \(MFA\) in AWS \(p. 91\)](#).

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": { "AWS": "arn:aws:iam::123456789012:root" },  
            "Action": "sts:AssumeRole",  
            "Condition": { "Bool": { "aws:MultiFactorAuthPresent": "true" } }  
        }  
    ]  
}
```

Important

If your `Principal` element contains an ARN that points to a specific IAM role or user, then that ARN is transformed to a unique principal ID when the policy is saved. This helps mitigate the risk of someone escalating their privileges by removing and recreating the role or user. You don't normally see this ID in the console, because there is also a reverse transformation back to the ARN when the trust policy is displayed. However, if you delete the role or user, then the relationship is broken. The policy no longer applies, even if you recreate the user or role because it does not match the principal ID stored in the trust policy. When this happens, the principal ID shows up in the console because AWS can no longer map it back to an ARN. The result is that if you delete and recreate a user or role referenced in a trust policy's `Principal` element, you must edit the role to replace the ARN. The user or role is transformed into the new principal ID when you save the policy.

The managed policy referenced in the second command is assumed to already exists in IAM. This policy allows the user who assumes the role to perform only the `ListBucket` action on an S3 bucket named `example_bucket`. The policy looks like this:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "s3>ListBucket",  
            "Resource": "arn:aws:s3:::example_bucket"  
        }  
    ]  
}
```

The commands to run are the following:

```
# Create the role and attach the trust policy that allows users in an account to switch to  
# the role.  
$ aws iam create-role --role-name Test-UserAccess-Role --assume-role-policy-document  
file://C:\policies\trustpolicyforacct123456789012.json  
  
# Attach the permissions policy (in this example a managed policy) to the role to specify  
# what it is allowed to do.
```

```
$ aws iam attach-role-policy --role-name Test-UserAccess-Role --policy-arn arn:aws:iam::123456789012:role/PolicyForRole
```

Important

Remember that this is only the first half of the configuration required. You must also give individual users in the trusted account permissions to switch to the role. For more information about this step, see [Granting a User Permissions to Switch Roles \(p. 205\)](#).

After you create the role and grant it permissions to perform AWS tasks or access AWS resources, the user can switch to the role. For more information, see [Switching to an IAM Role \(AWS Command Line Interface\) \(p. 211\)](#).

Creating an IAM Role (AWS API)

You can use API calls to create a role that an IAM user can switch to.

To create a role in code (API)

- Create a role: [CreateRole](#)
For the role's trust policy, you can specify a file location.
- Attach a managed permission policy to the role: [AttachRolePolicy](#)
-or-
Create an inline permission policy for the role: [PutRolePolicy](#)

Important

Remember that this is only the first half of the configuration required. You must also give individual users in the trusted account permissions to switch to the role. For more information about this step, see [Granting a User Permissions to Switch Roles \(p. 205\)](#).

After you create the role and grant it permissions to perform AWS tasks or access AWS resources, the user can switch to the role. For more information, see [Switching to an IAM Role \(API\) \(p. 214\)](#).

For more information about MFA, see [Using Multi-Factor Authentication \(MFA\) in AWS \(p. 91\)](#).

How to Use an External ID When Granting Access to Your AWS Resources to a Third Party

At times, you need to give a third party access to your AWS resources (delegate access). One important aspect of this scenario is the *External ID*, an optional piece of information that you can use in an IAM role trust policy to designate who can assume the role.

Important

AWS does not treat the external ID as a secret. After you create a secret like an access key pair or a password in AWS, you cannot view them again. The external ID for a role can be seen by anyone with permission to view the role.

For example, let's say that you decide to hire a third-party company called Example Corp to monitor your AWS account and help optimize costs. In order to track your daily spending, Example Corp needs to access your AWS resources. Example Corp also monitors many other AWS accounts for other customers.

Do not give Example Corp access to an IAM user and its long-term credentials in your AWS account. Instead, use an IAM role and its temporary security credentials. An IAM role provides a mechanism to allow a third party to access your AWS resources without needing to share long-term credentials (for example, an IAM user's access key).

You can use an IAM role to establish a trusted relationship between your AWS account and the Example Corp account. After this relationship is established, a member of the Example Corp account can call the

AWS STS [AssumeRole](#) API to obtain temporary security credentials. The Example Corp members can then use the credentials to access AWS resources in your account.

Note

For more information about the AssumeRole and other AWS API operations that you can call to obtain temporary security credentials, see [Requesting Temporary Security Credentials \(p. 233\)](#).

Here's a more detailed breakdown of this scenario:

1. You hire Example Corp, so they create a unique customer identifier for you. They give you your unique customer ID and their AWS account number. You need this information to create an IAM role in the next step.

Note

Example Corp can use any string value they want for the ExternalId, as long as it is unique for each customer. It can be a customer account number or even a random string of characters, as long as no two customers have the same value. It is not intended to be a 'secret'. Example Corp must provide the ExternalId value to each customer. What is key is that it must be generated by Example Corp and **not** their customers.

2. You sign in to AWS and create an IAM role that gives Example Corp access to your resources. Like any IAM role, the role has two policies, a permission policy and a trust policy. The role's trust policy specifies who can assume the role. In our sample scenario, the policy specifies the AWS account number of Example Corp as the `Principal`. This allows identities from that account to assume the role. In addition, you add an `Condition` element to the trust policy. This Condition tests the `ExternalID` context key to ensure that it matches the unique customer ID from Example Corp. For example:

```
"Principal": {"AWS": "Example Corp's AWS Account ID"},  
"Condition": {"StringEquals": {"sts:ExternalId": "Unique ID Assigned by Example Corp"}}
```

3. The permission policy for the role specifies what the role allows someone to do. For example, you could specify that the role allows someone to manage only your Amazon EC2 and Amazon RDS resources but not your IAM users or groups. In our sample scenario, you use the permission policy to give Example Corp read-only access to all of the resources in your account.
4. After you create the role, you provide the Amazon Resource Name (ARN) of the role to Example Corp.
5. When Example Corp needs to access your AWS resources, someone from the company calls the AWS `sts:AssumeRole` API. The call includes the ARN of the role to assume and the `ExternalID` parameter that corresponds to your customer ID.

If the request comes from someone using Example Corp's AWS account, and if the role ARN and the external ID are correct, the request succeeds. It then provides temporary security credentials that Example Corp can use to access the AWS resources that your role allows.

In other words, when a role policy includes an external ID, anyone who wants to assume the role must be a principal in the role and must include the correct external ID.

Why Do You Need to Use an External ID?

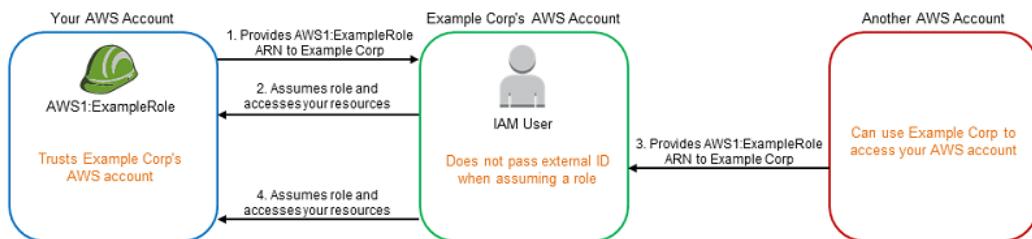
In abstract terms, the external ID allows the user that is assuming the role to assert the circumstances in which they are operating. It also provides a way for the account owner to permit the role to be assumed only under specific circumstances. The primary function of the external ID is to address and prevent the "confused deputy" problem.

The Confused Deputy Problem

To continue the previous example, Example Corp requires access to certain resources in your AWS account. But in addition to you, Example Corp has other customers and needs a way to access each customer's AWS resources. Instead of asking its customers for their AWS account access keys, which are

secrets that should never be shared, Example Corp requests a role ARN from each customer. But another Example Corp customer might be able to guess or obtain your role ARN. That customer could then use your role ARN to gain access to your AWS resources by way of Example Corp. This form of privilege escalation is known as the confused deputy problem.

The following diagram illustrates the confused deputy problem.



This diagram assumes the following:

- **AWS1** is your AWS account.
- **AWS1:ExampleRole** is a role in your account. This role's trust policy trusts Example Corp by specifying Example Corp's AWS account as the one that can assume the role.

Here's what happens:

1. When you start using Example Corp's service, you provide the ARN of **AWS1:ExampleRole** to Example Corp.
2. Example Corp uses that role ARN to obtain temporary security credentials to access resources in your AWS account. In this way, you are trusting Example Corp as a "deputy" that can act on your behalf.
3. Another AWS customer also starts using Example Corp's service, and this customer also provides the ARN of **AWS1:ExampleRole** for Example Corp to use. Presumably the other customer learned or guessed the **AWS1:ExampleRole**, which isn't a secret.
4. When the other customer asks Example Corp to access AWS resources in (what it claims to be) its account, Example Corp uses **AWS1:ExampleRole** to access resources in your account.

This is how the other customer could gain unauthorized access to your resources. Because this other customer was able to trick Example Corp into unwittingly acting on your resources, Example Corp is now a "confused deputy."

How Does the External ID Prevent the Confused Deputy Problem?

You address the confused deputy problem by including the `ExternalID` condition check in the role's trust policy. The "deputy" company inserts a unique external ID value for each customer into the request for AWS credentials. The external ID is a customer ID value that must be unique among Example Corp's customers and is out of the control of Example Corp's customers. This is why you get it from Example Corp and you don't come up with it on your own. This helps prevent one customer from successfully impersonating another customer. Example Corp always inserts the customer's assigned external ID, so you should never see a request coming from Example Corp with any external ID except your own.

In our scenario, imagine Example Corp's unique identifier for you is "12345," and its identifier for the other customer is "67890." These identifiers are simplified for this scenario. Generally, these identifiers are GUIDs. Assuming that these identifiers are unique among Example Corp's customers, they are sensible values to use for the external ID.

Example Corp gives the external ID value of "12345" to you. You must then add a `Condition` element to the role's trust policy that requires the `sts:ExternalID` value to be 12345, like this:

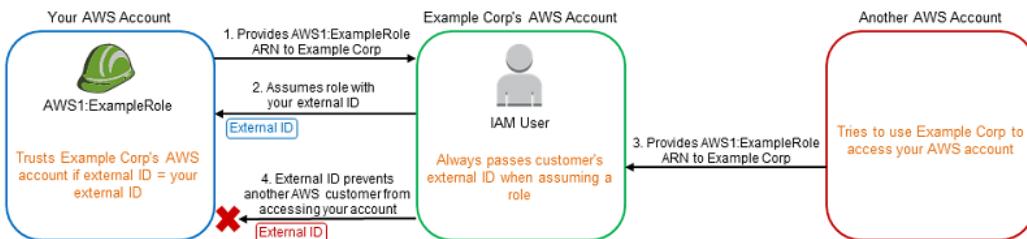
```
{
```

```

    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"AWS": "Example Corp's AWS Account ID"},
            "Condition": {"StringEquals": {"sts:ExternalId": "12345"}}
        }
    ]
}

```

The Condition element in this policy allows Example Corp to assume the role only when the AssumeRole API call includes the external ID value of "12345". Example Corp makes sure that whenever it assumes a role on behalf of a customer, it always includes that customer's external ID value in the AssumeRole call. Even if another customer supplies Example Corp with your ARN, it cannot know the external ID that Example Corp includes in its request to AWS. This helps prevent an unauthorized customer from gaining access to your resources, as shown in the following diagram.



1. As before, when you start using Example Corp's service, you provide the ARN of **AWS1:ExampleRole** to Example Corp.
2. When Example Corp uses that role ARN to assume the role **AWS1:ExampleRole**, Example Corp includes your external ID ("12345") in the AssumeRole API call. The external ID matches the role's trust policy, so the AssumeRole API call succeeds and Example Corp obtains temporary security credentials to access resources in your AWS account.
3. Another AWS customer also starts using Example Corp's service, and as before, this customer also provides the ARN of **AWS1:ExampleRole** for Example Corp to use.
4. But this time, when Example Corp attempts to assume the role **AWS1:ExampleRole**, it provides the external ID associated with the other customer ("67890"). The other customer has no way to change this. Example Corp does this because the request to use the role came from the other customer, so "67890" indicates the circumstance in which Example Corp is acting. Because you added a condition with your own external ID ("12345") to the trust policy of **AWS1:ExampleRole**, the AssumeRole API call fails. The other customer is prevented from gaining unauthorized access resources in your account (indicated by the red "X" in the diagram).

The external ID helps prevent any other customer from tricking Example Corp into unwittingly accessing your resources—it mitigates the confused deputy problem.

When Should I Use the External ID?

Use an external ID in the following situations:

- You are an AWS account owner and you have configured a role for a third party that accesses other AWS accounts in addition to yours. You should ask the third party for an external ID that it includes when it assumes your role. Then you check for that external ID in your role's trust policy. Doing so ensures that the external party can assume your role only when it is acting on your behalf.
- You are in the position of assuming roles on behalf of different customers like Example Corp in our previous scenario. You should assign a unique external ID to each customer and instruct them to add the external ID to their role's trust policy. You must then ensure that you always include the correct external ID in your requests to assume roles.

You probably already have a unique identifier for each of your customers, and this unique ID is sufficient for use as an external ID. The external ID is not a special value that you need to create explicitly, or track separately, just for this purpose.

You should always specify the external ID in your `AssumeRole` API calls. In addition when a customer gives you a role ARN, test whether you can assume the role both with and without the correct external ID. If you can assume the role without the correct external ID, don't store the customer's role ARN in your system. Wait until your customer has updated the role trust policy to require the correct external ID. In this way you help your customers to do the right thing, which helps to keep both of you protected against the confused deputy problem.

Creating a Role to Delegate Permissions to an AWS Service

Many AWS services require that you use roles to control what that service can access. A role that a service assumes to perform actions on your behalf is called a [service role \(p. 135\)](#). When a role serves a specialized purpose for a service, it is categorized as a [service role for EC2 instances \(p. 136\)](#) (for example), or a [service-linked role \(p. 136\)](#). See the [AWS documentation](#) for each service to see if it uses roles and to learn how to assign a role for the service to use.

For information about how roles help you to delegate permissions, see [Roles Terms and Concepts \(p. 135\)](#).

Creating a Role for an AWS Service (Console)

You can use the AWS Management Console to create a role for a service. Because some services support more than one service role, see the [AWS documentation](#) for your service to see which use case to choose. You can learn how to assign the necessary trust and permissions policies to the role so that the service can assume the role on your behalf.

To create a role for an AWS service (console)

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 of the IAM console, choose **Roles**, and then choose **Create role**.
3. Choose the **AWS Service** role type, and then choose the service that you want to allow to assume this role.
4. Choose the use case for your service. If the specified service has only one use case, it is selected for you. Use cases are defined by the service to include the trust policy that the service requires. Then choose **Next: Permissions**.
5. Choose one or more permissions policies to attach to the role. Depending on the use case that you selected, the service might do any of the following:
 - Define the permissions that the role uses
 - Allow you to choose from a limited set of permissions
 - Allow you to choose from any permissions
 - Allow you to select no policies at this time, create the policies later, and then attach them to the role

Select the box next to the policy that assigns the permissions that you want the users to have, and then choose **Next: Review**.

Note

The permissions that you specify are available to any entity that uses the role. By default, a role has no permissions.

6. For **Role name**, the degree of role name customization is defined by the service. If the service defines the role's name, this option is not editable. In other cases, the service might define a prefix for the role and allow you to type an optional suffix. Some services allow you to specify the entire name of your role.

If possible, type a role name or role name suffix to help you identify the purpose of this role. Role names must be unique within your AWS account. They are not distinguished by case. For example, you cannot create roles named both **PRODROLE** and **prodrole**. Because various entities might reference the role, you cannot edit the name of the role after it has been created.

7. (Optional) For **Role description**, type a description for the new role.
8. Review the role and then choose **Create role**.

Creating a Role for a Service (AWS CLI)

Creating a role from the AWS CLI involves multiple steps. When you use the console to create a role, many of the steps are done for you, but with the AWS CLI you must explicitly perform each step yourself. You must create the policy and assign a permission policy to the role. If the service you are working with is Amazon EC2, then you must also create an instance profile and add the role to it.

To create a role for an AWS service from the AWS CLI, use the following commands:

1. Create a role: [aws iam create-role](#)
2. Attach a managed permission policy to the role: [aws iam attach-role-policy](#)
or

Create an inline permission policy for the role: [aws iam put-role-policy](#)

If you are going to use the role with Amazon EC2 or another AWS service that uses Amazon EC2, you must store the role in an instance profile. An instance profile is a container for a role that can be attached to an Amazon EC2 instance when launched. An instance profile can contain only one role, and that limit cannot be increased. If you create the role using the AWS Management Console, the instance profile is created for you with the same name as the role. For more information about instance profiles, see [Using Instance Profiles \(p. 218\)](#). For information about how to launch an EC2 instance with a role, see [Controlling Access to Amazon EC2 Resources](#) in the *Amazon EC2 User Guide for Linux Instances*.

To create an instance profile and add a role to it from the AWS CLI, use the following commands:

- Create an instance profile: [aws iam create-instance-profile](#)
- Add the role to the instance profile: [aws iam add-role-to-instance-profile](#)

The following example shows all four steps. The example assumes that you are running on a client computer running Windows and have already configured your command line interface with your account credentials and region. For more information, see [Configuring the AWS Command Line Interface](#).

The example trust policy referenced in the first command contains the following JSON code to enable the Amazon EC2 service to assume the role.

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Principal": {"Service": "ec2.amazonaws.com"},  
        "Action": "sts:AssumeRole"  
    }  
}
```

The example permission policy referenced in the second command allows the role to perform only the `ListBucket` action on an S3 bucket named `example_bucket`. To learn how to create an IAM policy using this example JSON policy document, see [the section called “Create a Policy on the JSON Tab” \(p. 320\)](#).

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "s3>ListBucket",  
            "Resource": "arn:aws:s3:::example_bucket"  
        }  
    ]  
}
```

The AWS CLI commands to run for this example are the following:

```
# Create the role and attach the trust policy that enables EC2 to assume this role.  
$ aws iam create-role --role-name Test-Role-for-EC2 --assume-role-policy-document file:///  
trustpolicyforec2.json  
  
# Embed the permissions policy (in this example an inline policy) to the role to specify  
what it is allowed to do.  
$ aws iam put-role-policy --role-name Test-Role-for-EC2 --policy-name Permissions-Policy-  
For-Ec2 --policy-document file://permissionspolicyforec2.json  
  
# Create the instance profile required by EC2 to contain the role  
$ aws iam create-instance-profile --instance-profile-name EC2-ListBucket-S3  
  
# Finally, add the role to the instance profile  
$ aws iam add-role-to-instance-profile --instance-profile-name EC2-ListBucket-S3 --role-  
name Test-Role-for-EC2
```

When you launch the EC2 instance, specify the instance profile name in the [Configure Instance Details](#) page if you use the AWS console. If you use the `aws ec2 run-instances` CLI command, specify the `--iam-instance-profile` parameter.

Creating a Role for a Service (AWS API)

You can use the AWS API to create a service role.

To create a service role in code (API)

Use the following commands:

- Create a role: [CreateRole](#)
For the role's trust policy, you can specify a file location.
- Attach a managed permission policy to the role: [AttachRolePolicy](#)
or
Create an inline permission policy for the role: [PutRolePolicy](#)

If you are going to use the role with Amazon EC2 or another AWS service that uses Amazon EC2, you must store the role in an instance profile. An instance profile is a container for a role. Each instance profile can contain only one role, and that limit cannot be increased. If you create the role in the AWS Management Console, the instance profile is created for you with the same name as the role. For more information about instance profiles, see [Using Instance Profiles \(p. 218\)](#). For information about how to launch an Amazon EC2 instance with a role, see [Controlling Access to Amazon EC2 Resources](#) in the [Amazon EC2 User Guide for Linux Instances](#).

- Create an instance profile: [CreateInstanceProfile](#)
- Add the role to the instance profile: [AddRoleToInstanceProfile](#)

Creating a Role for a Third-Party Identity Provider (Federation)

Identity federation provides access to AWS resources to users by means of a third-party identity provider (IdP). To set up identity federation, you configure the provider and then create an IAM role that determines what permissions a federated user will have. For more information about federation and identity providers, see [Identity Providers and Federation \(p. 143\)](#).

Creating a Role for Federated Users (AWS Management Console)

The steps for creating a role for federated users depends on your choice of third-party providers:

- To create a role for users federated with a Web Identity or OpenID Connect (OIDC) IdP, see [Creating a Role for Web Identity or OpenID Connect Federation \(Console\) \(p. 196\)](#).
- To create a role for users federated with a SAML 2.0 IdP, see [Creating a Role for SAML 2.0 Federation \(Console\) \(p. 200\)](#).

Creating a Role for Federated Access (AWS Command Line Interface)

The steps to create a role for the supported identity providers (OIDC or SAML) from the AWS CLI are identical;—the difference is in the contents of the trust policy that you create in the prerequisite steps. Begin by following the steps in the **Prerequisites** section for the type of provider you are using:

- For an OIDC provider, follow the steps in [Prerequisites for an OIDC provider \(p. 196\)](#).
- For a SAML provider, follow the steps in [Prerequisites for a SAML provider \(p. 200\)](#).

Creating a role from the AWS CLI involves multiple steps. When you use the console to create a role, many of the steps are done for you, but with the CLI you must explicitly perform each step yourself. You must create the trust policy first, create the role, and then assign an permission policy to the role.

To create a role using the AWS CLI

Use the following commands:

- To create a role: `aws iam create-role`
- To attach a permission policy to the role:

`aws iam attach-role-policy` to attach an existing managed policy

or

`aws iam put-role-policy` to create an inline policy

The following example shows all the steps in a simple environment. The example assumes that you are running the AWS CLI on a computer running Windows, and have already configured the AWS CLI with your credentials. For more information, see [Configuring the AWS Command Line Interface](#).

The commands to run are the following:

```
# Create the role and attach the trust policy that enables users in an account to assume
# the role.
$ aws iam create-role --role-name Test-CrossAcct-Role --assume-role-policy-document file:///
trustpolicyforcognitofederation.json
```

```
# Attach the permissions policy to the role to specify what it is allowed to do.  
aws iam put-role-policy --role-name Test-CrossAcct-Role --policy-name Perms-Policy-For-  
CognitoFederation --policy-document file://permspolicyforcognitofederation.json
```

Creating a Role for Federated Access (Tools for Windows PowerShell)

The steps to create a role for the supported identity providers (OIDC or SAML) is identical; the difference is in the contents of the trust policy you create in the prerequisite steps. Follow the steps in the [Prerequisites](#) section for the type of provider you are using:

- For an OIDC provider, follow the steps in [Prerequisites for an OIDC provider \(p. 196\)](#).
- For a SAML provider, follow the steps in [Prerequisites for a SAML provider \(p. 200\)](#).

Creating a role using the Tools for Windows PowerShell involves multiple steps. When you use the console to create a role, many of the steps are done for you, but with the Tools for Windows PowerShell you must explicitly perform each step yourself. You must create the trust policy first, create the role, and then assign a permission policy to the role.

To create a role using the Tools for Windows PowerShell

Use the following commands:

- To create a role: [New-IAMRole](#)
- To attach a permission policy to the role:

[Register-IAMRolePolicy](#) to attach an existing managed policy

-or-

[Write-IAMRolePolicy](#) to create an inline policy

The following example shows all of the steps in a simple environment. The example assumes that you have already configured the Tools for Windows PowerShell with your credentials. For more information, see [Using AWS Credentials](#).

The commands to run are the following:

```
# Create the role and attach the trust policy that enables users in an account to assume  
the role.  
PS C:\> New-IAMRole -RoleName Test-Federation-Role -AssumeRolePolicyDocument (Get-Content -  
Raw C:\policies\trustpolicyforfederation.json)  
  
# Attach a managed permissions policy to the role to specify what it is allowed to do.  
PS C:\> Register-IAMRolePolicy -RoleName Test-Federation-Role -PolicyArn  
arn:aws:iam::aws:policy/PolicyForFederation
```

Creating a Role for Federated Access (IAM API)

Before you create the role, you must follow the steps in the Prerequisites section for the type of provider you are using:

- For an OIDC provider, follow the steps in [Prerequisites for an OIDC provider \(p. 196\)](#).
- For a SAML provider, follow the steps in [Prerequisites for a SAML provider \(p. 200\)](#).

To create a role for identity federation using the IAM API

Use the following commands:

- To create a role: [CreateRole](#)
- To attach a permission policy to the role:

[AttachRolePolicy](#) to attach an existing managed policy

or

[PutRolePolicy](#) to create an inline policy

Creating a Role for Web Identity or OpenID Connect Federation (Console)

Before you can create a role for web identity federation, you must first complete the following prerequisite steps.

To prepare to create a role for web identity federation

1. Begin by signing up as a developer with an IdP (or more than one). You also configure your app with the provider; when you do, the provider gives you an application or audience ID that's unique to your app. (Providers use different terminology for this process. This guide uses the term *configure* for the process of identifying your app with the provider.) Each provider gives you an app ID that's unique to that provider, so if you configure the same app with multiple providers, your app will have multiple app IDs. You can configure multiple apps with each provider. The following external links provide information about using one of the identity providers:
 - [Login with Amazon Developer Center](#)
 - [Add Facebook Login to Your App or Website](#) on the Facebook developers site.
 - [Using OAuth 2.0 for Login \(OpenID Connect\)](#) on the Google developers site.
2. After getting the required information from the identity provider, you can create an *identity provider* in IAM. For more information, see [Creating OpenID Connect \(OIDC\) Identity Providers \(p. 153\)](#).
3. Prepare the policies for the role that the IdP-authenticated users will assume. As with any role, a role for the mobile app contains two policies. One is the trust policy that specifies who can assume the role (the trusted entity or principal). The other policy (the permission policy) specifies the actual AWS actions and resources that the mobile app is allowed or denied access to (similar to user or resource policies).

For web identity providers, we recommend that you use [Amazon Cognito](#) to manage identities. In that case, the trust policy for the role must include a Statement similar to the following:

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Principal": {"Federated": "cognito-identity.amazonaws.com"},  
        "Action": "sts:AssumeRoleWithWebIdentity",  
        "Condition": {  
            "StringEquals": {"cognito-identity.amazonaws.com:aud": "us-east-2:12345678-abcd-abcd-abcd-123456"},  
            "ForAnyValue:StringLike": {"cognito-identity.amazonaws.com:amr": "unauthenticated"}  
        }  
    }  
}
```

Replace `us-east-2:12345678-abcd-abcd-abcd-123456` with the actual identity pool ID that Amazon Cognito assigned to you.

If you manually configure a web identity IdP, then the trust policy must grant an `Allow` effect for the `sts:AssumeRoleWithWebIdentity` action. In this role, you use two values that ensure that only your application can assume the role:

- For the `Principal` element, use the string `{"Federated":providerUrl/providerArn}`.
- For some common OpenID Connect (OIDC) IdPs, the `providerUrl` is the fully qualified URL. The following are acceptable ways to specify the principal for some common IdPs:

```
"Principal": {"Federated": "cognito-identity.amazonaws.com"}
```

```
"Principal": {"Federated": "www.amazon.com"}
```

```
"Principal": {"Federated": "graph.facebook.com"}
```

```
"Principal": {"Federated": "accounts.google.com"}
```

- For other OIDC providers, use the ARN of the OIDC identity provider you created in [Step 2](#), like the following example:

```
"Principal": {"Federated": "arn:aws:iam::123456789012:oidc-provider/server.example.com"}
```

- In the `Condition` element, use a `StringEquals` condition to limit permissions. Test the identity pool ID (for Amazon Cognito) or the app ID (for other providers). It should match the app ID that you got when you configured the app with the IdP. This ensures that the request is coming from your app. Test the app ID that you have against the following policy variables depending on the IdP that you are using:

```
"Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud": "us-east-1:12345678-ffff-ffff-ffff-123456"}}
```

```
"Condition": {"StringEquals": {"www.amazon.com:app_id": "amzn1.application-oa2-123456"}}
```

```
"Condition": {"StringEquals": {"graph.facebook.com:app_id": "111222333444555"}}
```

```
"Condition": {"StringEquals": {"accounts.google.com:aud": "66677788899900pro0"}}
```

For OIDC providers, use the fully qualified URL of the OIDC IdP with the `aud` context key, like the following example:

```
"Condition": {"StringEquals": {"server.example.com:aud": "appid_from_oidc_idp"}}
```

Notice that the values for the principal in the role are specific to an IdP. A role can specify only one principal. Therefore, if the mobile app allows users to sign in from more than one IdP, you must create a role for each IdP that you want to support.

Note

Because the policy for the trusted entity uses [policy variables](#) that represent the IdP and the app ID, you must set the policy's `Version` element to `2012-10-17` or a later supported version.

The following example shows a trust policy for a role designed for a mobile app if the user signs in from Login with Amazon. In the example, `amzn1.application-oa2-123456` represents the app ID that Amazon assigned when you configured the app using Login with Amazon.

```
{
    "Version": "2012-10-17",
    "Id": "RoleForLoginWithAmazon",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"Federated": "www.amazon.com"},
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {"StringEquals": {"www.amazon.com:app_id": "amzn1.application-
oa2-123456"}}
        }
    ]
}
```

The following example shows a policy for a role that the mobile app could assume if the user has signed in from Facebook. In this example, **111222333444555** represents the app ID assigned by Facebook. To learn how to create an IAM policy using this example JSON policy document, see [the section called "Create a Policy on the JSON Tab" \(p. 320\)](#).

```
{
    "Version": "2012-10-17",
    "Id": "RoleForFacebook",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"Federated": "graph.facebook.com"},
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {"StringEquals": {"graph.facebook.com:app_id": "111222333444555"}}
        }
    ]
}
```

The following example shows a policy for a role that the mobile app could assume if the user has signed in from Google. In this example, **666777888999000** represents the app ID assigned by Google. To learn how to create an IAM policy using this example JSON policy document, see [the section called "Create a Policy on the JSON Tab" \(p. 320\)](#).

```
{
    "Version": "2012-10-17",
    "Id": "RoleForGoogle",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"Federated": "accounts.google.com"},
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {"StringEquals": {"accounts.google.com:aud": "666777888999000"}}
        }
    ]
}
```

The following example shows a policy for a role that the mobile app could assume if the application is using Amazon Cognito. In this example, **us-east:12345678-ffff-ffff-ffff-123456** represents the identity pool ID assigned by Amazon Cognito. To learn how to create an IAM policy using this example JSON policy document, see [the section called "Create a Policy on the JSON Tab" \(p. 320\)](#).

```
{
    "Version": "2012-10-17",
    "Id": "RoleForCognito",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"Federated": "cognito-identity.amazonaws.com"},
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud": "us-
east:12345678-ffff-ffff-ffff-123456"}}
        }
    ]
}
```

```
    }]  
}
```

After you complete the prerequisites, you can create the role itself. The following procedure describes how to create the role for web identity/OIDC federation in the AWS Management Console. To create a role using the AWS CLI, Tools for Windows PowerShell, or AWS API, see the procedures at [Creating a Role for a Third-Party Identity Provider \(Federation\) \(p. 194\)](#).

To create an IAM role for web identity federation

If you are using Amazon Cognito, you should use the Amazon Cognito console to set up the roles. Otherwise, use the IAM console to create a role for web identity federation.

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, click **Roles** and then click **Create role**.
3. Choose the **Web identity** role type.
4. In the **Identity provider** list, choose the identity provider that you're creating the role for:
 - If you're creating a role for an individual web identity provider, choose **Login with Amazon**, **Facebook**, or **Google**.

Note

You must create a separate role for each identity provider that you want to support.

- Choose **Amazon Cognito** if you're creating a role for Amazon Cognito.

Note

You only need to manually create a role for use with Amazon Cognito when you are working on an advanced scenario. Otherwise, Amazon Cognito can create roles for you for standard scenarios. For more information about Amazon Cognito, see [Amazon Cognito Identity](#) in the [AWS Mobile SDK for iOS Developer Guide](#) and [Amazon Cognito Identity](#) in the [AWS Mobile SDK for Android Developer Guide](#).

5. Type the identifier for your application. The name identifier setting changes depending on which provider you choose:
 - If you're creating a role for Login with Amazon, type the app ID into the **Application ID** box.
 - If you're creating a role for Amazon Cognito, type the ID of the identity pool that you have created for your Amazon Cognito applications into the **Identity Pool ID** box.
 - If you're creating a role for Facebook, type the app ID into the **Application ID** box.
 - If you're creating a role for Amazon Cognito, type the audience name into the **Audience** box.
6. (Optional) Click **Add condition (optional)** to create additional conditions that must be met before users of your application can use the permissions that the role grants. For example, you can add a condition that grants access to AWS resources only for a specific IAM user ID.
7. Review your web identity information and then choose **Next: Permissions**.
8. Choose one or more permissions policies to attach to the role. By default, roles have no permissions. Select the box next to the policy that assigns the permissions that you want the web identity users to have. Then choose **Next: Review**.
9. For **Role name**, type a role name that helps you identify the purpose of this role. Role names must be unique within your AWS account. They are not distinguished by case. For example, you cannot create roles named both **PRODROLE** and **prodrole**. Because various entities might reference the role, you cannot edit the name of the role after it has been created.
10. (Optional) For **Role description**, type a description for the new role.
11. Review the role and then choose **Create role**.

Creating a Role for SAML 2.0 Federation (Console)

With identity federation, you can provide access to AWS resources for users who sign in from a third-party identity provider (IdP). To configure identity federation, you configure the provider and then you create an IAM role that determines what permissions a federated user has. For more information about federation and identity providers, see [Identity Providers and Federation \(p. 143\)](#).

Before you can create a role for SAML 2.0 federation, you must first complete the following prerequisite steps:

To prepare to create a role for SAML 2.0 federation

1. Before you create a role for SAML-based federation, you must create a SAML provider in IAM. For more information, see [Creating SAML Identity Providers \(p. 158\)](#).
2. Prepare the policies for the role that the SAML 2.0–authenticated users will assume. As with any role, a role for the SAML federation contains two policies. One is the trust policy that specifies who can assume the role (the trusted entity, or principal). The other policy (the permission policy) specifies the actual AWS actions and resources that the federated user is allowed or denied access to (similar to a user or resource policy).

For SAML 2.0 providers, the policy must include a `Statement` element similar to the following:

The trust policy must grant an `Allow` effect for the `sts:AssumeRoleWithSAML` action. In this role, you use two values that ensure that the role can be assumed only by your application:

- For the `Principal` element, use the string `{"Federated": "ARNofIdentityProvider"}`. Replace `ARNofIdentityProvider` with the ARN of the [SAML identity provider \(p. 149\)](#) that you created in Step 1.
- For the `Condition` element, use a `StringEquals` condition to test that the `saml:aud` attribute from the SAML response matches the SAML federation endpoint for AWS.

Note

Because the policy for the trusted entity uses [policy variables](#) that represent values in the SAML response, you must set the policy's `Version` element to `2012-10-17` or a later supported version.

The following example shows a trust policy for a role designed for a SAML federated user:

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Action": "sts:AssumeRoleWithSAML",  
        "Principal": {"Federated": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:saml-provider/PROVIDER-NAME"},  
        "Condition": {"StringEquals": {"SAML:aud": "https://signin.aws.amazon.com/saml"}}  
    }  
}
```

Replace the principal ARN with the actual ARN for the SAML provider that you created in IAM. It will have your own account ID and the actual provider name.

After completing the prerequisite steps, you can create the role itself.

To create a role for SAML-based federation

1. Make sure you've created a SAML provider in IAM, as described in [About SAML 2.0-based Federation \(p. 149\)](#).
2. In the navigation pane of the console, choose **Roles** and then choose **Create role**.
3. Choose the **SAML 2.0 federation** role type.
4. In the **SAML Provider** list, select the provider that you're creating the role for.
5. Choose the SAML 2.0 access level method. Choose **Allow programmatic access only** to create a role that can be assumed programmatically from the AWS API. Then choose **Allow programmatic and AWS Management Console access** to create a role that can be assumed programmatically and from the console. The roles created by both are similar, but the role that can also be assumed from the console includes a trust policy with a particular condition. That condition explicitly ensures that the SAML audience (`SAML:aud` attribute) is set to the AWS sign-in endpoint for SAML (`https://signin.aws.amazon.com/saml`).
6. If you're creating a role for programmatic access, select an attribute from the **Attribute** list. Then in the **Value** box, type a value to include in the role. This restricts role access to users from the identity provider whose SAML authentication response (assertion) includes the attributes that you specify. You must specify at least one attribute to ensure that your role is limited to a subset of users at your organization.

If you're creating a role for programmatic and console access, the `SAML:aud` attribute is automatically added and set to the URL of the AWS SAML endpoint (`https://signin.aws.amazon.com/saml`).

7. To add more attribute-related conditions to the trust policy, choose **Add condition (optional)**, select the additional condition, and specify a value.

Note

The list displays the most commonly used SAML attributes. IAM supports additional attributes that you can use to create conditions. (For a list of the supported attributes, see [Available Keys for SAML Federation](#) in the topic [IAM JSON Policy Elements Reference \(p. 426\)](#).) If you need a condition for a supported SAML attribute that's not in the list, you can manually add that condition by editing the trust policy after you create the role.

8. Review your SAML 2.0 trust information and then choose **Next: Permissions**.
9. Choose one or more permissions policies to attach to the role. By default, roles have no permissions. Select the box next to the policy that assigns the permissions that you want the federated users to have. Then choose **Next: Review**.
10. For **Role name**, type a role name that helps you identify the purpose of this role. Role names must be unique within your AWS account. They are not distinguished by case. For example, you cannot create roles named both `PRODROLE` and `prodrole`. Because various entities might reference the role, you cannot edit the name of the role after it has been created.
11. (Optional) For **Role description**, type a description for the new role.
12. Review the role and then choose **Create role**.

After you create the role, you complete the SAML trust by configuring your identity provider software with information about AWS and the roles that you want your federated users to use. This is referred to as configuring relying party trust between your IdP and AWS. For more information, see [Configuring your SAML 2.0 IdP with Relying Party Trust and Adding Claims \(p. 160\)](#).

Examples of Policies for Delegating Access

The following examples show how you can allow or grant an AWS account access to the resources in another AWS account. To learn how to create an IAM policy using these example JSON policy documents, see the section called "[Create a Policy on the JSON Tab](#)" (p. 320).

Topics

- [Using Roles to Delegate Access to Another AWS Account's Resources \(p. 202\)](#)
- [Using a Policy to Delegate Access To Services \(p. 202\)](#)
- [Using a Resource-Based Policy to Delegate Access to an Amazon S3 Bucket in Another Account \(p. 202\)](#)
- [Using a Resource-Based Policy to Delegate Access to an Amazon SQS Queue in Another Account \(p. 203\)](#)
- [Cannot Delegate Access When the Account is Denied Access \(p. 204\)](#)

Using Roles to Delegate Access to Another AWS Account's Resources

For a tutorial that shows how to use IAM roles to grant users in one account access to AWS resources that are in another account, see [Tutorial: Delegate Access Across AWS Accounts Using IAM Roles \(p. 26\)](#).

Important

You can include the ARN for a specific role or user in the `Principal` element of a role trust policy. When you save the policy, AWS transforms the ARN to a unique principal ID. This helps mitigate the risk of someone escalating their privileges by removing and recreating the role or user. You don't normally see this ID in the console, because there is also a reverse transformation back to the ARN when the trust policy is displayed. However, if you delete the role or user, then the relationship is broken. The policy no longer applies, even if you recreate the user or role because it does not match the principal ID stored in the trust policy. When this happens, the principal ID shows up in the console because AWS can no longer map it back to an ARN. The result is that if you delete and recreate a user or role referenced in a trust policy's `Principal` element, you must edit the role to replace the ARN. It is transformed into the new principal ID when you save the policy.

Using a Policy to Delegate Access To Services

The following example shows a policy that can be attached to a role. The policy enables two services, Amazon EMR and AWS Data Pipeline, to assume the role. The services can then perform any tasks granted by the permissions policy assigned to the role (not shown). To specify multiple service principals, you do not specify two `Service` elements; you can have only one. Instead, you use an array of multiple service principals as the value of a single `Service` element.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "Service": [  
                    "elasticmapreduce.amazonaws.com",  
                    "datapipeline.amazonaws.com"  
                ]  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

Using a Resource-Based Policy to Delegate Access to an Amazon S3 Bucket in Another Account

In this example, account A uses a resource-based policy (an Amazon S3 [bucket policy](#)) to grant account B full access to account A's S3 bucket. Then account B creates an IAM user policy to delegate that access to account A's bucket to one of the users in account B.

The S3 bucket policy in account A might look like the following policy. In this example, account A's S3 bucket is named *mybucket*, and account B's account number is 111122223333. It does not specify any individual users or groups in account B, only the account itself.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AccountBAccess1",
            "Effect": "Allow",
            "Principal": {"AWS": "111122223333"},
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::mybucket",
                "arn:aws:s3:::mybucket/*"
            ]
        }
    ]
}
```

Alternatively, account A can use Amazon S3 [Access Control Lists \(ACLs\)](#) to grant account B access to an S3 bucket or a single object within a bucket. In that case, the only thing that changes is how account A grants access to account B. Account B still uses a policy to delegate access to an IAM group in account B, as described in the next part of this example. For more information about controlling access on S3 buckets and objects, go to [Access Control](#) in the *Amazon Simple Storage Service Developer Guide*.

The administrator of account B might create the following policy sample. The policy allows read access to a group or user in account B. The preceding policy grants access to account B. However, individual groups and users in account B cannot access the resource until a group or user policy explicitly grants permissions to the resource. The permissions in this policy can only be a subset of those in the preceding cross-account policy. Account B cannot grant more permissions to its groups and users than account A granted to account B in the first policy. In this policy, the `Action` element is explicitly defined to allow only `List` actions, and the `Resource` element of this policy matches the `Resource` for the bucket policy implemented by account A.

To implement this policy account B uses IAM to attach it to the appropriate user (or group) in account B.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3>List*",
            "Resource": [
                "arn:aws:s3:::mybucket",
                "arn:aws:s3:::mybucket/*"
            ]
        }
    ]
}
```

Using a Resource-Based Policy to Delegate Access to an Amazon SQS Queue in Another Account

In the following example, account A has an Amazon SQS queue that uses a resource-based policy attached to the queue to grant queue access to account B. Then account B uses an IAM group policy to delegate access to a group in account B.

The following example queue policy gives account B permission to perform the `SendMessage` and `ReceiveMessage` actions on account A's queue named *queue1*, but only between noon and 3:00 p.m. on November 30, 2014. Account B's account number is 1111-2222-3333. Account A uses Amazon SQS to implement this policy.

```
{
```

```

    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"AWS": "111122223333"},
            "Action": [
                "sns:SendMessage",
                "sns:ReceiveMessage"
            ],
            "Resource": ["arn:aws:sns:*:123456789012:queue1"],
            "Condition": {
                "DateGreaterThan": {"aws:CurrentTime": "2014-11-30T12:00Z"},
                "DateLessThan": {"aws:CurrentTime": "2014-11-30T15:00Z"}
            }
        }
    ]
}

```

Account B's policy for delegating access to a group in account B might look like the following example. Account B uses IAM to attach this policy to a group (or user).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sns:*",
            "Resource": "arn:aws:sns:*:123456789012:queue1"
        }
    ]
}
```

In the preceding IAM user policy example, account B uses a wildcard to grant its user access to all Amazon SNS actions on account A's queue. However account B can delegate access only to the extent that account B has been granted access. The account B group that has the second policy can access the queue only between noon and 3:00 p.m. on November 30, 2014. The user can only perform the SendMessage and ReceiveMessage actions, as defined in account A's Amazon SNS queue policy.

Cannot Delegate Access When the Account is Denied Access

An AWS account cannot delegate access to another account's resources if the other account has explicitly denied access to the user's parent account. The deny propagates to the users under that account whether or not the users have existing policies granting them access.

For example, account A writes a bucket policy on account A's S3 bucket that explicitly denies account B access to account A's bucket. But account B writes an IAM user policy that grants a user in account B access to account A's bucket. The explicit deny applied to account A's S3 bucket propagates to the users in account B. It overrides the IAM user policy granting access to the user in account B. (For detailed information how permissions are evaluated, see [IAM JSON Policy Evaluation Logic \(p. 458\)](#).)

Account A's bucket policy might look like the following policy. In this example, account A's S3 bucket is named *mybucket*, and account B's account number is 1111-2222-3333. Account A uses Amazon S3 to implement this policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AccountBDeny",
            "Effect": "Deny",
            "Principal": {"AWS": "111122223333"},
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::mybucket/*"
        }
    ]
}
```

This explicit deny overrides any policies in account B that provide permission to access the S3 bucket in account A.

Using IAM Roles

Before an IAM user, application, or service can use a role that you created, you must grant permissions to switch to the role. You can use any policy attached to one of an IAM user's groups or to the user itself to grant the necessary permissions. This section describes how to grant users permission to use a role, and then how the user can switch to a role using the AWS Management Console, the Tools for Windows PowerShell, the AWS Command Line Interface (AWS CLI) and the [AssumeRole API](#).

Important

If you create a role programmatically instead of in the IAM console, then you have an option to add a Path of up to 512 characters in addition to the RoleName, which can be up to 64 characters long. However, if you intend to use a role with the Switch Role feature in the AWS console, then the combined Path and RoleName cannot exceed 64 characters.

Note

When using the AWS Management Console, you can switch roles only when signed in as an IAM user. You cannot switch roles when signed in as the AWS account root user.

Topics

- [Granting a User Permissions to Switch Roles \(p. 205\)](#)
- [Granting a User Permissions to Pass a Role to an AWS Service \(p. 207\)](#)
- [Switching to a Role \(AWS Management Console\) \(p. 209\)](#)
- [Switching to an IAM Role \(AWS Command Line Interface\) \(p. 211\)](#)
- [Switching to an IAM Role \(Tools for Windows PowerShell\) \(p. 212\)](#)
- [Switching to an IAM Role \(API\) \(p. 214\)](#)
- [Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances \(p. 215\)](#)
- [Revoking IAM Role Temporary Security Credentials \(p. 220\)](#)

Granting a User Permissions to Switch Roles

When you [create a role for cross-account access \(p. 184\)](#), you establish trust from the account that owns the role and the resources (trusting account) to the account that contains the users (trusted account). To do this, you specify the trusted account number as the Principal in the role's trust policy. That allows *potentially* any user in the trusted account to assume the role. To complete the configuration, the administrator of the trusted account must give specific groups or users in that account permission to switch to the role.

To grant a user permission to switch to a role, you create a new policy for the user or edit an existing policy to add the required elements. You can then send the users a link that takes the user to the **Switch Role** page with all the details already filled in. Alternatively, you can provide the user with the account ID number or account alias that contains the role and the role name. The user then goes to the **Switch Role** page and adds the details manually. For details on how a user switches roles, see [Switching to a Role \(AWS Management Console\) \(p. 209\)](#).

Note that you can switch roles only when you sign in as an IAM user. You cannot switch roles when you sign in as the AWS account root user.

Important

You cannot switch roles in the AWS Management Console to a role that requires an [ExternalID \(p. 187\)](#) value. You can switch to such a role only by calling the AssumeRole API that supports the ExternalID parameter.

Notes

- This topic discusses policies for a *user*, because we are ultimately granting permissions to a user to accomplish a task. However, it is [best practice not to grant permissions directly to an individual user \(p. 44\)](#). For easier management, we recommend assigning policies and granting permissions to IAM groups and then making the users members of the appropriate groups.
- When you switch roles in the AWS Management Console, the console always uses your original credentials to authorize the switch. This applies whether you sign in as an IAM user, as a SAML-federated role, or as a web-identity federated role. For example, if you switch to RoleA, it uses your original user or federated role credentials to determine if you are allowed to assume RoleA. If you then try to switch to RoleB *while you are using RoleA*, it still uses your **original** user or federated role credentials to authorize your attempt to switch to RoleB, not the credentials for RoleA.

Creating or Editing the Policy

A policy that grants a user permission to assume a role must include a statement with the `Allow` effect on the `sts:AssumeRole` action and the Amazon Resource Name (ARN) of the role in a `Resource` element, as shown in the following example. Users that get the policy (either through group membership or directly attached) are allowed to switch to the specified role.

Note

Note that if `Resource` is set to `*`, the user can assume any role in any account that trusts the user's account (the role's trust policy specifies the user's account as `Principal`). As a best practice, we recommend that you follow the [principle of least privilege](#) and specify the complete ARN for only the role(s) that the user needs.

The following example shows a policy that lets the user assume roles in only one account and additionally specifies by wildcard (*) that the user can only switch to a role in that account if the role name begins with the letters "Test" followed by any other combination of characters.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "sts:AssumeRole",  
            "Resource": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:role/Test*"  
        }  
    ]  
}
```

Note

The permissions that the role grants to the user do not add to the permissions already granted to the user. When a user switches to a role, the user temporarily gives up his or her original permissions in exchange for those granted by the role. When the user exits the role, then the original user permissions are automatically restored. For example, if the user's permissions allow working with Amazon EC2 instances, but the role's permissions policy does not grant those permissions, then while using the role the user cannot work with Amazon EC2 instances in the console, and temporary credentials obtained via `AssumeRole` do not work with Amazon EC2 instances programmatically.

Providing Information to the User

After you create a role and grant your user permissions to switch to it, you must provide the user with the role name and the account ID number or account alias that contains the role. You can make things easier for your users by sending them a link that is preconfigured with the account ID and role name. You can see the role link on the final page of the **Create Role** wizard or in the **Role Summary** page for any cross-account enabled role.

Note

If you create the role with the AWS CLI , Tools for Windows PowerShell, or the AWS API, then you can create the role with a *path* in addition to a name. If you do so, then you must provide the complete path and role name to your users to type on the **Switch Role** page of the AWS Management Console. For example: division_abc/subdivision_efg/role_XYZ.

Important

If you create the role programmatically instead of in the IAM console, then you have an option to add a Path of up to 512 characters in addition to the RoleName, which can be up to 64 characters long. However, to use a role with the Switch Role feature in the AWS console, the combined Path and RoleName cannot exceed 64 characters.

You can also use the following format to manually construct the link. Substitute your account ID or alias and the role name for the two parameters in the request:

```
https://signin.aws.amazon.com/switchrole?  
account=YourAccountIDOrAliasHere&roleName=pathIfAny/YourRoleNameHere
```

We recommend that you direct your users to the topic [Switching to a Role \(AWS Management Console\) \(p. 209\)](#) to step them through the process.

Note

For security purposes, you can use AWS CloudTrail to audit role switching. If CloudTrail is turned on for the account, IAM logs actions that are performed with the role's temporary security credentials. For more information, see [CloudTrail Event Reference](#) in the *AWS CloudTrail User Guide*.

Granting a User Permissions to Pass a Role to an AWS Service

To configure many AWS services, you must *pass* an IAM role to the service that defines what that service can do on your behalf. For example, to provide applications running on an Amazon EC2 instance with AWS credentials, you pass a role to EC2 to use with the instance that provides those credentials. You define what the credentials allow the applications running on the instance to do by attaching an IAM policy that grants the required permissions to the role.

To pass a role (and its permissions) to an AWS service, a user must have permissions to *pass the role* to the service. This helps administrators ensure that only approved users can configure a service with a role that grants permissions. To allow a user to pass a role to an AWS service, you must grant the `PassRole` permission to the user's IAM user, role, or group.

A user can pass a role ARN as a parameter in any API operation that uses the role to assign permissions to the service. The service then checks whether that user has the `iam:PassRole` permission. To limit the user to passing only approved roles, you can filter the `iam:PassRole` permission with the `Resources` element of the IAM policy statement.

Example 1

Imagine that you want to grant a user the ability to pass any of an approved set of roles to the Amazon EC2 service upon launching an instance. You need three elements:

- An IAM *permissions policy* attached to the role that determines what the role can do. Scope permissions to only the actions that the role must perform, and to only the resources that the role needs for those actions. You can use AWS managed or customer-created IAM permissions policy.

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",
```

```
        "Action": [ "A list of the permissions the role is allowed to use" ],
        "Resource": [ "A list of the resources the role is allowed to access" ]
    }
```

- A *trust policy* for the role that allows the service to assume the role. For example, you could attach the following trust policy to the role with the `UpdateAssumeRolePolicy` action. This trust policy allows Amazon EC2 to use the role and the permissions attached to the role.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TrustPolicyStatementThatAllowsEC2ServiceToAssumeTheAttachedRole",
            "Effect": "Allow",
            "Principal": { "Service": "ec2.amazonaws.com" },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

- An IAM *permissions policy* attached to the IAM user that allows the user to pass only those policies that are approved. `iam:PassRole` usually is accompanied by `iam:GetRole` so that the user can get the details of the role to be passed. In this example, the user can pass only roles that exist in the specified account with names that begin with `EC2-roles-for-XYZ-`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::<account-id>:role/EC2-roles-for-XYZ-*"
        }
    ]
}
```

Now the user can start an Amazon EC2 instance with an assigned role. Applications running on the instance can access temporary credentials for the role through the instance profile metadata. The permission policies attached to the role determine what the instance can do.

Example 2

Amazon Relational Database Service (Amazon RDS) supports a feature called Enhanced Monitoring. This feature enables Amazon RDS to monitor a database instance using an agent. It also allows Amazon RDS to log metrics to Amazon CloudWatch Logs. To enable this feature, you must create a service role to give Amazon RDS permissions to monitor and write metrics to your logs.

To create a role for Amazon RDS Enhanced Monitoring

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Choose **Roles**, and then choose **Create role**.
3. Choose the **AWS Service** role type, and then choose the **Amazon RDS Role for Enhanced Monitoring** service. Then choose **Next: Permissions**.
4. Choose the **AmazonRDSEnhancedMonitoringRole** permissions policy and then choose **Next: Review**.
5. For **Role name**, type a role name that helps you identify the purpose of this role. Role names must be unique within your AWS account. They are not distinguished by case. For example, you cannot

create roles named both **PRODROLE** and **prodrole**. Because various entities might reference the role, you cannot edit the name of the role after it has been created.

6. (Optional) For **Role description**, type a description for the new role.
7. Review the role and then choose **Create role**.

The role automatically gets a trust policy that grants the `monitoring.rds.amazonaws.com` service permissions to assume the role. After it does, Amazon RDS can perform all of the actions that the `AmazonRDSEnhancedMonitoringRole` policy allows.

The user that you want to enable Enhanced Monitoring needs a policy that includes a statement that allows the user to pass the role, like the following. Use your account number and replace the role name with the name you provided in step 3:

```
{  
    "Sid": "PolicyStatementToAllowUserToPassOneSpecificRole",  
    "Effect": "Allow",  
    "Action": [ "iam:PassRole" ],  
    "Resource": "arn:aws:iam:::role/RDS-Monitoring-Role"  
}
```

You can combine this statement with statements in another policy or put it in its own policy. To instead specify that the user can pass any role that begins with `RDS-`, you can replace the role name in the resource ARN with a wildcard, for example:

```
"Resource": "arn:aws:iam:::role/RDS-*"
```

Switching to a Role (AWS Management Console)

A *role* specifies a set of permissions that you can use to access AWS resources that you need. In that sense, it is similar to a [user in AWS Identity and Access Management \(IAM\)](#). When you sign in as a user, you get a specific set of permissions. However, you don't sign in to a role, but once signed in you can switch to a role. This temporarily sets aside your original user permissions and instead gives you the permissions assigned to the role. The role can be in your own account or any other AWS account. For more information about roles, their benefits, and how to create them, see [IAM Roles \(p. 135\)](#), and [Creating IAM Roles \(p. 184\)](#).

Important

When you switch roles in the AWS Management Console, the console always uses your original credentials to authorize the switch. This applies whether you sign in as an IAM user, as a SAML-federated role, or as a web-identity federated role. For example, if you switch to RoleA, it uses your original user or federated role credentials to determine if you are allowed to assume RoleA. If you then try to switch to RoleB *while you are using RoleA*, it still uses your **original** user or federated role credentials to authorize your attempt to switch to RoleB, not the credentials for RoleA.

This section describes how to use the IAM console to switch to a role:

- You can only switch roles when you sign in as an IAM user. You cannot switch roles if you sign in as the AWS account root user.
- If your administrator provides you with a link, click the link and then skip to step [Step 5](#) in the following procedure. The link takes you to the appropriate web page and fills in the account ID (or alias) and the role name for you.

Tip

You can manually construct the link yourself by using the following format:

`https://signin.aws.amazon.com/switchrole?
account=account_id_number&roleName=role_name&displayName=text_to_display`

Where your administrator provides the *account_id_number* and *role_name* to you. For *text_to_display*, see the explanation in step 5 in the following procedure.

Important

If you create the role programmatically instead of in the IAM console, then you have an option to add a Path of up to 512 characters in addition to the RoleName, which can be up to 64 characters long. However, to use a role with the Switch Role feature in the AWS console, the combined Path and RoleName cannot exceed 64 characters.

- You can manually switch roles using the information your administrator provides by using the procedures below.

To troubleshoot common issues that you might encounter when you assume a role, see [I Can't Assume a Role \(p. 400\)](#).

To switch to a role

1. Sign in to the AWS Management Console as an IAM user and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the IAM console, choose your user name on the navigation bar in the upper right. It typically looks like this: *username@account_ID_number_or_alias*.
3. For **Identity**, select **Switch Role**. If this is the first time selecting this option, a page appears with more information. After reading it, click **Switch Role**. If you clear your browser cookies, this page can appear again.
4. On the **Switch Role** page, type the account ID number or the account alias and the name of the role that was provided by your administrator.

Note

If your administrator created the role with a path, such as *division_abc/subdivision_efg/roleToDoXYZ*, then you must type that complete path and name in the **Role** box. If you type only the role name, the attempt to switch role fails.

Important

If you create the role programmatically instead of in the IAM console, then you have an option to add a Path of up to 512 characters in addition to the RoleName, which can be up to 64 characters long. However, to use a role with the Switch Role feature in the IAM console, the combined Path and RoleName cannot exceed 64 characters. This is a limit of the browser cookies that store the role name.

5. (Optional) Type text that you want to appear on the navigation bar in place of your user name when this role is active. A name is suggested, based on the account and role information, but you can change it to whatever has meaning for you. You can also select a color to highlight the display name. The name and color can help remind you when this role is active, which changes your permissions. For example, for a role that gives you access to the test environment, you might specify a **Display Name of Test** and select the green **Display Color**. For the role that gives you access to production, you might specify a **Display Name of Production** and select red as the **Display Color**.
6. Click **Switch Role**. The display name and color replace your user name on the navigation bar, and you can start using the permissions that the role grants you.

Tip

The last several roles that you used appear on the **Identity** menu. The next time you need to switch to one of those roles, you can simply click the desired role. You only need to enter the account and role information manually if the role is not displayed on the Identity menu.

To stop using a role

1. In the IAM console, select your role's **Display Name** on the right side of the navigation bar.

2. Select **Back to *UserName***. The role and its permissions are deactivated, and the permissions associated with your IAM user and groups are automatically restored.

Switching to an IAM Role (AWS Command Line Interface)

A *role* specifies a set of permissions that you can use to access AWS resources that you need. In that sense, it is similar to a [user in AWS Identity and Access Management \(IAM\)](#). When you sign in as a user, you get a specific set of permissions. However, you don't sign in to a role, but once signed in as a user you can switch to a role. This temporarily sets aside your original user permissions and instead gives you the permissions assigned to the role. The role can be in your own account or any other AWS account. For more information about roles, their benefits, and how to create and configure them, see [IAM Roles \(p. 135\)](#), and [Creating IAM Roles \(p. 184\)](#).

Important

The permissions of your IAM user and any roles that you switch to are not cumulative. Only one set of permissions is active at a time. When you switch to a role, you temporarily give up your user permissions and work with the permissions that are assigned to the role. When you exit the role, your user permissions are automatically restored.

You can run an AWS CLI command using a role only when you are signed in as an IAM user, as an [externally authenticated user \(p. 143\)](#) ([SAML \(p. 149\)](#) or [OIDC \(p. 143\)](#)) already using a role, or when run from within an Amazon EC2 instance that is attached to a role through its instance profile. You cannot switch to a role when you are signed in as the AWS account root user.

This section describes how to switch roles when you work at the command line with the AWS Command Line Interface.

Imagine that you have an IAM user for working in the development environment and you occasionally need to work with the production environment at the command line with the [AWS CLI](#). You already have an access key credential set available to you. This can be the access key pair assigned to your standard IAM user; or, if you signed-in as a federated user, it can be the access key pair for the role initially assigned to you. If your current permissions grant you the ability to assume a specific role, then you can identify that role in a "profile" in the AWS CLI configuration files. That command is then run with the permissions of the specified role, not the original identity. Note that when you specify that profile, and thus use the new role, in an AWS CLI command, you cannot make use of your original permissions in the development account at the same time because only one set of permissions can be in effect at a time.

Note

For security purposes, you can use AWS CloudTrail to audit the use of roles in the account. To identify a role's actions in CloudTrail logs, you can use the role session name. When the AWS CLI assumes a role on a user's behalf as described in this topic, a role session name is automatically created as `AWS-CLI-session-nnnnnnnn`, where `nnnnnnnn` is an integer that represents the time in [Unix epoch time](#) (the number of seconds since midnight UTC on January 1, 1970). For more information, see [CloudTrail Event Reference](#) in the [AWS CloudTrail User Guide](#).

To switch to a role using the AWS CLI

1. If you have never used the AWS CLI, then you must first configure your default CLI profile. Open a command prompt and set up your AWS CLI installation to use the access key from your IAM user or from your federated role. For more information, see [Configuring the AWS Command Line Interface](#) in the [AWS Command Line Interface User Guide](#).

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-east-2
Default output format [None]: json
```

2. Create a new profile for the role in the .aws/config file. The following example creates a profile called "prodaccess" that switches to the role *ProductionAccessRole* in the 123456789012 account. You get the role ARN from the account administrator who created the role. When this profile is invoked, the AWS CLI uses the credentials of the *source_profile* to request credentials for the role. Because of that, the identity referenced as the *source_profile* must have *sts:AssumeRole* permissions to the role specified in the *role_arn*.

```
[profile prodaccess]
role_arn = arn:aws:iam::123456789012:role/ProductionAccessRole
source_profile = default
```

3. After you create the new profile, any AWS CLI command that specifies the parameter --profile prodaccess runs under the permissions attached to the IAM role ProductionAccessRole instead of the default user.

```
$ aws iam list-users --profile prodaccess
```

This command works if the permissions assigned to the *ProductionAccessRole* enable listing the users in the current AWS account.

4. To return to the permissions granted by your original credentials, run commands without the --profile parameter. The AWS CLI reverts to using the credentials in your default profile, which you configured in [Step 1](#).

For more information, see [Assuming a Role in the AWS Command Line Interface User Guide](#).

Switching to an IAM Role (Tools for Windows PowerShell)

A *role* specifies a set of permissions that you can use to access AWS resources that you need. In that sense, it is similar to a [user in AWS Identity and Access Management \(IAM\)](#). When you sign in as a user, you get a specific set of permissions. However, you don't sign in to a role, but once signed in you can switch to a role. This temporarily sets aside your original user permissions and instead gives you the permissions assigned to the role. The role can be in your own account or any other AWS account. For more information about roles, their benefits, and how to create and configure them, see [IAM Roles \(p. 135\)](#), and [Creating IAM Roles \(p. 184\)](#).

Important

The permissions of your IAM user and any roles that you switch to are not cumulative. Only one set of permissions is active at a time. When you switch to a role, you temporarily give up your user permissions and work with the permissions that are assigned to the role. When you exit the role, your user permissions are automatically restored.

This section describes how to switch roles when you work at the command line with the AWS Tools for Windows PowerShell.

Imagine that you have an account in the development environment and you occasionally need to work with the production environment at the command line using the [Tools for Windows PowerShell](#). You already have one access key credential set available to you. These can be an access key pair assigned to your standard IAM user; or, if you signed-in as a federated user, they can be the access key pair for the role initially assigned to you. You can use these credentials to run the `Use-STSRole` cmdlet that passes the ARN of a new role as a parameter. The command returns temporary security credentials for the requested role. You can then use those credentials in subsequent PowerShell commands with the role's permissions to access resources in production. While you use the role, you cannot make use of your user privileges in the Development account because only one set of permissions can be in effect at a time.

Note

For security purposes, you can use AWS CloudTrail to audit the use of roles in the account. The cmdlet `Use-STSRole` must include a `-RoleSessionName` parameter with a value between

2 and 64 characters long that can include letters, numbers, and the =, .@- characters. The role session name identifies actions in CloudTrail logs that are performed with the temporary security credentials. For more information, see [CloudTrail Event Reference](#) in the *AWS CloudTrail User Guide*.

Note that all access keys and tokens are examples only and cannot be used as shown. Replace with the appropriate values from your live environment.

To switch to a role from the Tools for Windows PowerShell

1. Open a PowerShell command prompt and configure the default profile to use the access key from your current IAM user or from your federated role. If you have previously used the Tools for Windows PowerShell, then this is likely already done. Note that you can switch roles only if you are signed in as an IAM user, not the AWS account root user.

```
PS C:\> Set-AWSCredentials -AccessKey AKIAIOSFODNN7EXAMPLE -SecretKey wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY -StoreAs MyMainUserProfile
PS C:\> Initialize-AWSDefaults -ProfileName MyMainUserProfile -Region us-east-2
```

For more information, see [Using AWS Credentials](#) in the *AWS Tools for Windows PowerShell User Guide*.

2. To retrieve credentials for the new role, run the following command to switch to the `RoleName` role in the 123456789012 account. You get the role ARN from the account administrator who created the role. The command requires that you provide a session name as well. You can choose any text for that. The following command requests the credentials and then captures the `Credentials` property object from the returned results object and stores it in the `$creds` variable.

```
PS C:\> $creds = (Use-STSRole -RoleArn "arn:aws:iam::123456789012:role/RoleName" -RoleSessionName "MyRoleSessionName").Credentials
```

`$creds` is an object that now contains the `AccessKeyId`, `SecretAccessKey`, and `SessionToken` elements that you need in the following steps. The following sample commands illustrate typical values:

```
PS C:\> $creds.AccessKeyId
AKIAIOSFODNN7EXAMPLE

PS C:\> $creds.SecretAccessKey
wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

PS C:\> $creds.SessionToken
AQoDYXdzEGcaEXAMPLE2gsYULo+Im5ZEXAMPLEeYjs1M2FUiG1Jx9tQqNMBEXAMPLEcvSRyh0FW7jEXAMPLEW
+vE/7s1HRp
XviG7b+qYf4nD00EXAMPLEmj4wxS04L/uZEXAMPLEcihzFB5lTYLto9dyBgSDyEXAMPLE9/
g7QRUhZp4bqbEXAMPLEnWGPY
Oj59pFA41NKC1kVgkREXAMPLEjlzxQ7y52gekeVEXAMPLEDiB9ST3UuysgsKdEXAMPLE1TVastU1AOSKFEXAMPLEiywCC/
C
s8EXAMPLEpZgOs+6hz4AP4KEXAMPLERbASP+4eZScEXAMPLEsnf87eNhyDHq6ikBQ==

PS C:\> $creds.Expiration
Thursday, June 18, 2018 2:28:31 PM
```

3. To use these credentials for any subsequent command, include them with the `-Credentials` parameter. For example, the following command uses the credentials from the role and works only if the role is granted the `iam>ListRoles` permission and can therefore run the `Get-IAMRoles` cmdlet:

```
PS C:\> get-iamroles -Credential $Creds
```

4. To return to your original credentials, simply stop using the `-Credentials $Creds` parameter and allow PowerShell to revert to the credentials that are stored in the default profile.

Switching to an IAM Role (API)

A *role* specifies a set of permissions that you can use to access AWS resources that you need. In that sense, it is similar to a [user in AWS Identity and Access Management \(IAM\)](#). An application assumes a role to receive permissions to carry out required tasks and interact with AWS resources. The role can be in your own account or any other AWS account. For more information about roles, their benefits, and how to create and configure them, see [IAM Roles \(p. 135\)](#), and [Creating IAM Roles \(p. 184\)](#).

This section describes how to switch roles from within code that uses the AWS API.

Note that all access keys and tokens are examples only and cannot be used as shown. Replace with the appropriate values from your live environment.

To assume a role, an application calls the AWS STS [AssumeRole](#) API and passes the ARN of the role to use. The [AssumeRole](#) API returns a set of temporary security credentials that you can use in subsequent AWS API calls to access resources in the account that owns the role. The temporary credentials have whatever permissions are defined in the role's permission policy. The call to [AssumeRole](#) can optionally pass a supplemental policy that can further restrict (filter) the permissions of the temporary security credentials that the [AssumeRole](#) API returns. You can call [AssumeRole](#) only with IAM user or IAM role credentials. You cannot call [AssumeRole](#) with the credentials of the AWS account root user.

Note

For security purposes, you can use AWS CloudTrail to audit the use of roles in the account. The call to [AssumeRole](#) must include a role session name between 2 and 64 characters long that can include letters, numbers, and the =, .@- characters. The role session name is used in CloudTrail logs to identify actions performed by the temporary security credentials. For more information, see [CloudTrail Event Reference](#) in the *AWS CloudTrail User Guide*.

The following example in Python using the Boto3 interface to AWS ([AWS SDK for Python \(Boto V3\)](#)) shows how to call [AssumeRole](#) and how to use the temporary security credentials returned by [AssumeRole](#) to list all Amazon S3 buckets in the account that owns the role.

```
import boto3

# The calls to AWS STS AssumeRole must be signed with the access key ID
# and secret access key of an existing IAM user or by using existing temporary
# credentials such as those from another role. (You cannot call AssumeRole
# with the access key for the root account.) The credentials can be in
# environment variables or in a configuration file and will be discovered
# automatically by the boto3.client() function. For more information, see the
# Python SDK documentation:
# http://boto3.readthedocs.io/en/latest/reference/services/sts.html#client

# create an STS client object that represents a live connection to the
# STS service
sts_client = boto3.client('sts')

# Call the assume_role method of the STSConnection object and pass the role
# ARN and a role session name.
assumedRoleObject = sts_client.assume_role(
    RoleArn="arn:aws:iam::account-of-role-to-assume:role/name-of-role",
    RoleSessionName="AssumeRoleSession1"
)
```

```
# From the response that contains the assumed role, get the temporary
# credentials that can be used to make subsequent API calls
credentials = assumedRoleObject['Credentials']

# Use the temporary credentials that AssumeRole returns to make a
# connection to Amazon S3
s3_resource = boto3.resource(
    's3',
    aws_access_key_id = credentials['AccessKeyId'],
    aws_secret_access_key = credentials['SecretAccessKey'],
    aws_session_token = credentials['SessionToken'],
)

# Use the Amazon S3 resource object that is now configured with the
# credentials to access your S3 buckets.
for bucket in s3_resource.buckets.all():
    print(bucket.name)
```

Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances

Applications that run on an EC2 instance must include AWS credentials in their AWS API requests. You could have your developers store AWS credentials directly within the EC2 instance and allow applications in that instance to use those credentials. But developers would then have to manage the credentials and ensure that they securely pass the credentials to each instance and update each EC2 instance when it's time to rotate the credentials. That's a lot of additional work.

Instead, you can and should use an IAM role to manage *temporary* credentials for applications that run on an EC2 instance. When you use a role, you don't have to distribute long-term credentials (such as a user name and password or access keys) to an EC2 instance. Instead, the role supplies temporary permissions that applications can use when they make calls to other AWS resources. When you launch an EC2 instance, you specify an IAM role to associate with the instance. Applications that run on the instance can then use the role-supplied temporary credentials to sign API requests.

Using roles to grant permissions to applications that run on EC2 instances requires a bit of extra configuration. An application running on an EC2 instance is abstracted from AWS by the virtualized operating system. Because of this extra separation, an additional step is needed to assign an AWS role and its associated permissions to an EC2 instance and make them available to its applications. This extra step is the creation of an *instance profile* that is attached to the instance. The instance profile contains the role and can provide the role's temporary credentials to an application that runs on the instance. Those temporary credentials can then be used in the application's API calls to access resources and to limit access to only those resources that the role specifies. Note that only one role can be assigned to an EC2 instance at a time, and all applications on the instance share the same role and permissions.

Using roles in this way has several benefits. Because role credentials are temporary and rotated automatically, you don't have to manage credentials, and you don't have to worry about long-term security risks. In addition, if you use a single role for multiple instances, you can make a change to that one role and the change is propagated automatically to all the instances.

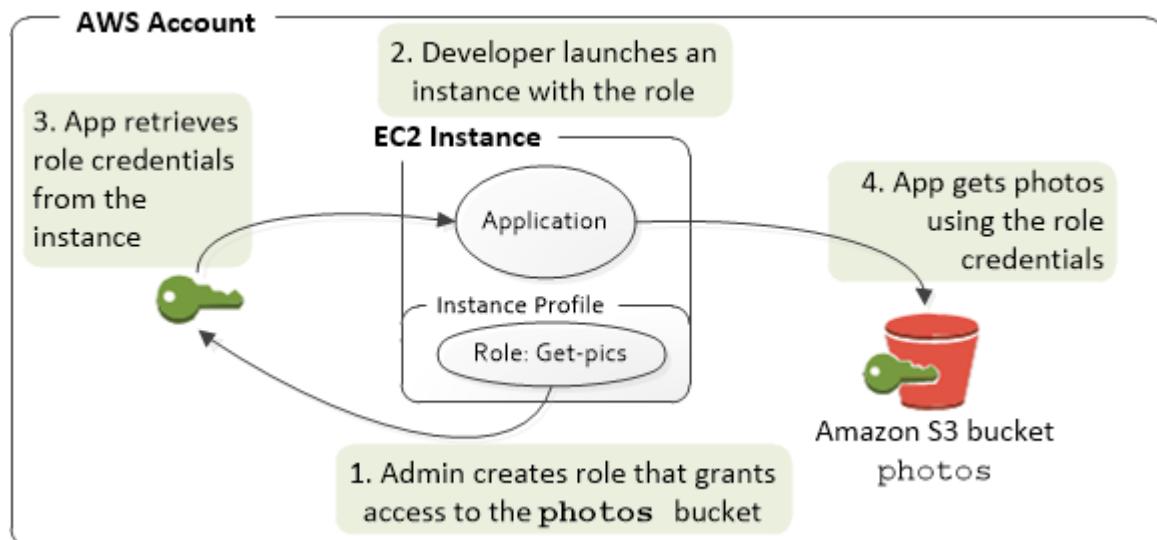
Note

Although a role is usually assigned to an EC2 instance when you launch it, a role can also be attached to an EC2 instance that is already running. To learn how to attach a role to a running instance, see [IAM Roles for Amazon EC2](#).

How Do Roles for EC2 Instances Work?

In the following figure, a developer runs an application on an EC2 instance that requires access to the S3 bucket named `photos`. An administrator creates the `Get-pics` role. The role includes policies that grant read permissions for the bucket and that allow the developer to launch the role with an EC2 instance.

When the application runs on the instance, it can use the role's temporary credentials to access the photos bucket. The administrator doesn't have to grant the developer permission to access the photos bucket, and the developer never has to share or manage credentials.



1. The administrator uses IAM to create the **Get-pics** role. In the role's trust policy, the administrator specifies that only EC2 instances can assume the role. In the role's permission policy, the administrator specifies read-only permissions for the photos bucket.
2. A developer launches an EC2 instance and assigns the **Get-pics** role to that instance.

Note

If you use the IAM console, the instance profile is managed for you and is mostly transparent to you. However, if you use the AWS CLI or API to create and manage the role and EC2 instance, then you must create the instance profile and assign the role to it as separate steps. Then, when you launch the instance, you must specify the instance profile name instead of the role name.

3. When the application runs, it obtains temporary security credentials from Amazon EC2 [instance metadata](#), as described in [Retrieving Security Credentials from Instance Metadata](#). These are [temporary security credentials](#) (p. 231) that represent the role and are valid for a limited period of time.

With some [AWS SDKs](#), the developer can use a provider that manages the temporary security credentials transparently. (The documentation for individual AWS SDKs describes the features supported by that SDK for managing credentials.)

Alternatively, the application can get the temporary credentials directly from the instance metadata of the EC2 instance. Credentials and related values are available from the `iam/security-credentials/role-name` category (in this case, `iam/security-credentials/Get-pics`) of the metadata. If the application gets the credentials from the instance metadata, it can cache the credentials.

4. Using the retrieved temporary credentials, the application accesses the photo bucket. Because of the policy attached to the **Get-pics** role, the application has read-only permissions.

The temporary security credentials that are available on the instance are automatically rotated before they expire so that a valid set is always available. The application just needs to make sure that it gets a new set of credentials from the instance metadata before the current ones expire. If the AWS SDK manages credentials, the application doesn't need to include additional logic to refresh the credentials. However, if the application gets temporary security credentials from the instance metadata and has cached them, it should get a refreshed set of credentials every hour, or at least

15 minutes before the current set expires. The expiration time is included in the information that is returned in the `iam/security-credentials/role-name` category.

Permissions Required for Using Roles with Amazon EC2

To launch an instance with a role, the developer must have permission to launch EC2 instances and permission to pass IAM roles.

The following sample policy allows users to use the AWS Management Console to launch an instance with a role. The policy includes wildcards (*) to allow a user to pass any role and to perform all Amazon EC2 actions. The `ListInstanceProfiles` action allows users to view all of the roles that are available in the AWS account.

Example policy that grants a user permission to use the Amazon EC2 console to launch an instance with any role

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": [  
            "iam:PassRole",  
            "iam>ListInstanceProfiles",  
            "ec2:*"  
        ],  
        "Resource": "*"  
    }]  
}
```

Restricting Which Roles Can Be Passed to EC2 Instances (Using PassRole)

You can use the `PassRole` permission to restrict which role a user can pass to an EC2 instance when the user launches the instance. This helps prevent the user from running applications that have more permissions than the user has been granted—that is, from being able to obtain elevated privileges. For example, imagine that user Alice has permissions only to launch EC2 instances and to work with Amazon S3 buckets, but the role she passes to an EC2 instance has permissions to work with IAM and Amazon DynamoDB. In that case, Alice might be able to launch the instance, log into it, get temporary security credentials, and then perform IAM or DynamoDB actions that she's not authorized for.

To restrict which roles a user can pass to an EC2 instance, you create a policy that allows the `PassRole` action. You then attach the policy to the user (or to an IAM group that the user belongs to) who will launch EC2 instances. In the `Resource` element of the policy, you list the role or roles that the user is allowed to pass to EC2 instances. When the user launches an instance and associates a role with it, Amazon EC2 checks whether the user is allowed to pass that role. Of course, you should also ensure that the role that the user can pass does not include more permissions than the user is supposed to have.

Note

`PassRole` is not an API action in the same way that `RunInstances` or `ListInstanceProfiles` is. Instead, it's a permission that AWS checks whenever a role ARN is passed as a parameter to an API (or the console does this on the user's behalf). It helps an administrator to control which roles can be passed by which users. In this case, it ensures that the user is allowed to attach a specific role to an Amazon EC2 instance.

Example policy that grants a user permission to launch an EC2 instance with a specific role

The following sample policy allows users to use the Amazon EC2 API to launch an instance with a role. The `Resource` element specifies the Amazon Resource Name (ARN) of a role. By specifying the ARN, the policy grants the user the permission to pass only the `Get-pics` role. If the user tries to specify a different role when launching an instance, the action fails.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "ec2:RunInstances",  
            "Resource": "*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": "iam:PassRole",  
            "Resource": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:role/Get-pics"  
        }  
    ]  
}
```

How Do I Get Started?

To understand how roles work with EC2 instances, you need to use the IAM console to create a role, launch an EC2 instance that uses that role, and then examine the running instance. You can examine the [instance metadata](#) to see how the role's temporary credentials are made available to an instance. You can also see how an application that runs on an instance can use the role. Use the following resources to learn more.

- SDK walkthroughs. The AWS SDK documentation includes walkthroughs that show an application running on an EC2 instance that uses temporary credentials for roles to read an Amazon S3 bucket. Each of the following walkthroughs presents similar steps with a different programming language:
 - [Using IAM Roles for EC2 Instances with the SDK for Java](#) in the *AWS SDK for Java Developer Guide*
 - [Using IAM Roles for EC2 Instances with the SDK for .NET](#) in the *AWS SDK for .NET Developer Guide*
 - [Using IAM Roles for EC2 Instances with the SDK for Ruby](#) in the *AWS SDK for Ruby Developer Guide*

The walkthroughs provide complete step-by-step instructions for creating and compiling the example program, creating the role, launching the instance, connecting to the instance, deploying the example program, and testing it.

Related Information

For more information about creating roles or roles for EC2 instances, see the following information:

- For more information about [using IAM roles with Amazon EC2 instances](#), go to the *Amazon EC2 User Guide for Linux Instances*.
- To create a role, see [Creating IAM Roles \(p. 184\)](#)
- For more information about using temporary security credentials, see [Temporary Security Credentials \(p. 231\)](#).
- If you work with the IAM API or CLI, you must create and manage IAM instance profiles. For more information about instance profiles, see [Using Instance Profiles \(p. 218\)](#).
- For more information about temporary security credentials for roles in the instance metadata, see [Retrieving Security Credentials from Instance Metadata](#) in the *Amazon EC2 User Guide for Linux Instances*.

Using Instance Profiles

An instance profile is a container for an IAM role that you can use to pass role information to an EC2 instance when the instance starts.

Managing Instance Profiles using the AWS Management Console

If you use the AWS Management Console to create a role for Amazon EC2, the console automatically creates an instance profile and gives it the same name as the role. When you then use the Amazon EC2 console to launch an instance with an IAM role, you can select a role to associate with the instance. In the console, the list that's displayed is actually a list of instance profile names. The console does not create an instance profile for a role that is not associated with Amazon EC2.

Managing Instance Profiles using the AWS CLI, Tools for Windows PowerShell, and AWS API

If you manage your roles from the AWS CLI, Tools for Windows PowerShell, or the AWS API, you create roles and instance profiles as separate actions. You can give the roles and instance profiles different names, so you have to know the names of your instance profiles as well as the names of roles they contain so that you can choose the correct instance profile when you launch an EC2 instance.

Note

An instance profile can contain only one IAM role, although a role can be included in multiple instance profiles. This limit of one role per instance cannot be increased. You can remove the existing role and then add a different role to an instance profile. You must then wait for the change to appear across all of AWS because of [eventual consistency](#). To force the change, you must [disassociate the instance profile](#) and then [associate the instance profile](#), or you can stop your instance and then restart it.

You can use the following commands to work with instance profiles in an AWS account.

Create an instance profile

- AWS CLI: `aws iam create-instance-profile`
- Tools for Windows PowerShell: `New-IAMInstanceProfile`
- AWS API: `CreateInstanceProfile`

Add a role to an instance profile

- AWS CLI: `aws iam add-role-to-instance-profile`
- Tools for Windows PowerShell: `Add-IAMRoleToInstanceProfile`
- AWS API: `AddRoleToInstanceProfile`

List instance profiles

- AWS CLI: `aws iam list-instance-profiles`, `aws iam list-instance-profiles-for-role`
- Tools for Windows PowerShell: `Get-IAMInstanceProfiles`
- AWS API: `ListInstanceProfiles`, `ListInstanceProfilesForRole`

Get information about an instance profile

- AWS CLI: `aws iam get-instance-profile`
- Tools for Windows PowerShell: `Get-IAMInstanceProfile`
- AWS API: `GetInstanceProfile`

Remove a role from an instance profile

- AWS CLI: `aws iam remove-role-from-instance-profile`
- Tools for Windows PowerShell: `Remove-IAMRoleFromInstanceProfile`

- AWS API: [RemoveRoleFromInstanceProfile](#)

Delete an instance profile

- AWS CLI: `aws iam delete-instance-profile`
- Tools for Windows PowerShell: [Remove-IAMInstanceProfile](#)
- AWS API: [DeleteInstanceProfile](#)

You can also attach a role to an already running EC2 instance by using the following commands. For more information, see [IAM Roles for Amazon EC2](#).

Attach an instance profile with a role to a stopped or running EC2 instance

- AWS CLI: `aws ec2 associate-iam-instance-profile`
- Tools for Windows PowerShell: [Register-Ec2IamInstanceProfile](#)
- AWS API: [AssociateIamInstanceProfile](#)

Get information about an instance profile attached to an EC2 instance

- AWS CLI: `aws ec2 describe-iam-instance-profile-associations`
- Tools for Windows PowerShell: [Get-EC2IamInstanceProfileAssociation](#)
- AWS API: [DescribeIamInstanceProfileAssociations](#)

Detach an instance profile with a role from a stopped or running EC2 instance

- AWS CLI: `aws ec2 disassociate-iam-instance-profile`
- Tools for Windows PowerShell: [Unregister-Ec2IamInstanceProfile](#)
- AWS API: [DisassociateIamInstanceProfile](#)

Revoking IAM Role Temporary Security Credentials

Warning

If you follow the steps on this page, all users with current sessions created by assuming the role are denied access to all AWS actions and resources. This can result in users losing unsaved work.

When you enable users to access the AWS Management Console with a long session duration time (such as 12 hours), their temporary credentials do not expire as quickly. If users inadvertently expose their credentials to an unauthorized third party, that party has access for the duration of the session. However, you can immediately revoke all permissions to the role's credentials issued before a certain point in time if you need to. All temporary credentials for that role issued before the specified time become invalid. This forces all users to reauthenticate and request new credentials.

Note

You cannot revoke the session for a [service-linked role \(p. 136\)](#).

When you revoke permissions for a role using the procedure in this topic, AWS attaches a new inline policy to the role that denies all permissions to all actions. It includes a condition that applies the restrictions only if the user assumed the role *before* the point in time when you revoke the permissions. If the user assumes the role *after* you revoked the permissions, then the deny policy does not apply to that user.

Important

This deny policy applies to all users of the specified role, not just those with longer duration console sessions.

Minimum Permissions to Revoke Session Permissions from a Role

To successfully revoke session permissions from a role, you must have the `AttachRolePolicy` permission for the role. This allows you to add the `AWSRevokeOlderSessions` policy to the role.

Revoking Session Permissions

To revoke the session permissions from a role, take the following steps:

To immediately deny all permissions to any current user of role credentials

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 of the **IAM Dashboard**, choose **Roles**, and then choose the name (not the check box) of the role whose permissions you want to revoke.
3. On the **Summary** page for the selected role, choose the **Revoke sessions** tab.
4. On the **Revoke sessions** tab, choose **Revoke active sessions**.
5. AWS asks you to confirm the action. Choose **Revoke active sessions** on the dialog box.

IAM immediately attaches a policy named `AWSRevokeOlderSessions` to the role. The policy denies all access to users who assumed the role before the moment you chose **Revoke active sessions**. Any user who assumes the role *after* you chose **Revoke active sessions** is **not** affected.

Important

When you update existing policy permissions, or when you apply a new policy to a user or a resource, it may take a few minutes for policy updates to take effect.

Note

Don't worry about remembering to delete the policy. Any user who assumes the role *after* you revoked sessions is not affected by the policy. If you choose to **Revoke Sessions** again later, then the date/time stamp in the policy is refreshed and it again denies all permissions to any user who assumed the role before the new specified time.

Valid users whose sessions are revoked in this way must acquire temporary credentials for a new session to continue working. Note that the AWS CLI caches credentials until they expire. To force the CLI to delete and refresh cached credentials that are no longer valid, run one of the following commands:

Linux, macOS, or Unix

```
$ rm -r ~/.aws/cli/cache
```

Windows

```
C:\> del /s /q %UserProfile%\.aws\cli\cache
```

For more information, see [Disabling Permissions for Temporary Security Credentials \(p. 253\)](#).

Managing IAM Roles

Occasionally you need to modify or delete the roles that you have created. To change a role you can modify the policies associated with the role, change who can access the role, and edit the permissions that the role grants to users. You can also delete roles that are no longer needed. You can manage your roles from the AWS Management Console, the AWS CLI, and the API.

Topics

- [Modifying a Role \(p. 222\)](#)
- [Deleting Roles or Instance Profiles \(p. 226\)](#)

Modifying a Role

You can change or modify a role in the following ways:

- To change who can assume a role, modify the role's trust policy.

Note

If the role is a [service-linked role \(p. 136\)](#), the role's trust policy cannot be modified. Service-linked roles appear with **(Service-linked role)** in the **Trusted entities** column of the table.

- To change the permissions allowed by the role, modify the role's permissions policy (or policies).

Note

If the role is a [service-linked role \(p. 136\)](#), the role's permissions can be modified only from the service that depends on the role. Service-linked roles appear with **(Service-linked role)** in the **Trusted entities** column of the table. See the [AWS documentation](#) for your service to see whether it supports this feature.

- To change the description of the role, modify the description text.

You can use the AWS Management Console, the [AWS Command Line Tools](#), the Tools for Windows PowerShell, or the IAM API to make these changes.

Modifying a Role (Console)

You can use the AWS Management Console to modify a role.

To change which trusted principals can access the role (console)

1. In the navigation pane of the IAM console, choose **Roles**.
2. In the list of roles in your account, choose the name of the role that you want to modify.
3. Choose the **Trust relationships** tab, and then choose **Edit trust relationship**.
4. Edit the trust policy as needed. To add additional trusted principals, specify them in the **Principal** element. Remember that policies are written in the **JSON** format, and JSON arrays are surrounded by square brackets [] and separated by commas. As an example, the following policy snippet shows how to reference two AWS accounts in the **Principal** element:

```
"Principal": {  
    "AWS": [  
        "arn:aws:iam::111122223333:root",  
        "arn:aws:iam::444455556666:root"  
    ]  
},
```

Remember that adding an account to the trust policy of a role is only half of establishing the trust relationship. By default, no users in the trusted accounts can assume the role until the administrator for that account grants the users the permission to assume the role. To do that, the administrator adds the Amazon Resource Name (ARN) of the role to an **Allow** element for the **sts:AssumeRole** action. For more information, see the next procedure and the topic [Granting a User Permissions to Switch Roles \(p. 205\)](#).

If your role can be used by one or more trusted services rather than AWS accounts, then the policy might contain an element similar to the following:

```
"Principal": {  
    "Service": [  
        "opsworks.amazonaws.com",  
        "ec2.amazonaws.com"  
    ]  
},
```

- When you are done editing, choose **Update Trust Policy** to save your changes.

For more information about policy structure and syntax, see [IAM Policies \(p. 275\)](#) and the [IAM JSON Policy Elements Reference \(p. 426\)](#).

To allow users in a trusted external account to use the role (console)

For more information and detail about this procedure, see [Granting a User Permissions to Switch Roles \(p. 205\)](#).

- Sign in to the trusted external AWS account.
- Decide whether to attach the permissions to a user or to a group. In the navigation pane of the IAM console, choose **Users** or **Groups** accordingly.
- Choose the name of the user or group to which you want to grant access, and then choose the **Permissions** tab.
- Do one of the following:
 - To edit a customer managed policy, choose the name of the policy, choose **Edit policy**, and then choose the **JSON** tab. You cannot edit an AWS managed policy. AWS managed policies appear with the AWS icon (). For more information about the difference between AWS managed policies and customer managed policies, see [Managed Policies and Inline Policies \(p. 279\)](#).
 - To edit an inline policy, choose the arrow next to the name of the policy and choose **Edit policy**.
- In the policy editor, add a new **Statement** element that specifies the following:

```
{  
    "Effect": "Allow",  
    "Action": "sts:AssumeRole",  
    "Resource": "arn:aws:iam::AWS account ID that contains the role:role/role name"  
}
```

Replace the values in red with the actual values from the ARN of the role in the original account that users in this trusted external account can use.

Remember that you can have only one **Statement** keyword. However, a statement can have several elements in an array, with elements separated by commas in their own curly braces {} and all of the elements surrounded by square brackets [].

- Follow the prompts on screen to finish editing the policy.

For more information about editing customer managed and inline policies in the AWS Management Console, see [Editing IAM Policies \(p. 337\)](#).

To change the permissions allowed by a role (console)

- In the navigation pane of the IAM console, choose **Roles**.
- Choose the name of the role to modify, and then choose the **Permissions** tab.
- Do one of the following:

- To edit an existing customer managed policy, choose the name of the policy and then choose **Edit policy**.

Note

You cannot edit an AWS managed policy. AWS managed policy appear with the AWS icon



() For more information about the difference between AWS managed policies and customer managed policies, see [Managed Policies and Inline Policies \(p. 279\)](#).

- To attach an existing managed policy, choose **Add permissions**.
- To edit an existing inline policy, choose the arrow next to the name of the policy and choose **Edit Policy**.
- To embed a new inline policy, choose **Add inline policy**.

For example policies that delegate permissions through roles, see [Example Policies \(p. 295\)](#).

For more information about permissions, see [IAM Policies \(p. 275\)](#).

To change the description of a role (console)

1. In the navigation pane of the IAM console, choose **Roles**.
2. Choose the name of the role to modify.
3. Next to **Role description** and on the far right, choose **Edit**.
4. Type a new description in the box and choose **Save**.

Modifying a Role (AWS CLI, AWS Tools for Windows PowerShell, AWS API)

You can use the AWS Command Line Interface or IAM API to modify a role.

To change the trusted principals that can access the role (AWS CLI, AWS Tools for Windows PowerShell, AWS API)

1. If you don't know the name of the role that you want to modify, use one of the following commands to list the roles in your account:
 - AWS CLI: `aws iam list-roles`
 - AWS Tools for Windows PowerShell: `Get-IAMRoles`
 - IAM API: `ListRoles`
2. (Optional) To view the current trust policy for a role, use one of the following commands:
 - AWS CLI: `aws iam get-role`
 - AWS Tools for Windows PowerShell: `Get-IAMRole`
 - IAM API: `GetRole`
3. To modify the trusted principals that can access the role, create a text file with the updated trust policy. You can use any text editor to construct the policy.

For example, the following policy snippet shows how to reference two AWS accounts in the `Principal` element:

```
"Principal": {  
    "AWS": [  
        "arn:aws:iam::111122223333:root",  
        "arn:aws:iam::444455556666:root"  
    ]  
}
```

```
},
```

Remember that adding an account to the trust policy of a role is only half of establishing the trust relationship. By default, no users in the trusted accounts can assume the role until the administrator for that account grants the users the permission to assume the role. To do this, the administrator must add the Amazon Resource Name (ARN) of the role to an `Allow` element for the `sts:AssumeRole` action. For more information, see the next procedure and the topic [Granting a User Permissions to Switch Roles \(p. 205\)](#).

4. To update the trust policy, use one of the following commands:

- AWS CLI: [aws iam update-assume-role-policy](#)
- AWS Tools for Windows PowerShell: [Update-IAMAssumeRolePolicy](#)
- IAM API: [UpdateAssumeRolePolicy](#)

To allow users in a trusted external account to use the role (AWS CLI, AWS Tools for Windows PowerShellAWS API)

For more information and detail about this procedure, see [Granting a User Permissions to Switch Roles \(p. 205\)](#).

1. Begin by creating a policy that grants permissions to assume the role. For example, the following policy contains the minimum necessary permissions:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "sts:AssumeRole",  
            "Resource": "arn:aws:iam::AWS account ID that contains the role:role/role name"  
        }  
    ]  
}
```

Create a JSON file that contains a policy similar to the preceding example. Replace the values in red with the actual values from the ARN of the role that users are allowed to assume. After you have created the policy, use one of the following commands to upload it to IAM:

- AWS CLI: [aws iam create-policy](#)
- AWS Tools for Windows PowerShell: [New-IAMPolicy](#)
- IAM API: [CreatePolicy](#)

The output of this command contains the ARN of the policy. Make a note of this ARN because you will need to use it in a later step.

2. Decide which user or group to attach the policy to. If you don't know the name of the user or group that you want to modify, use one of these commands to list the users or group in your account:
 - AWS CLI: [aws iam list-users](#) or [aws iam list-groups](#)
 - AWS Tools for Windows PowerShell: [Get-IAMUsers](#) or [Get-IAMGroups](#)
 - IAM API: [ListUsers](#) or [ListGroups](#)
3. Use one of the following commands to attach the policy that you created in a previous step to the user or group:
 - AWS CLI: [aws iam attach-user-policy](#) or [aws iam attach-group-policy](#)
 - AWS Tools for Windows PowerShell: [Register-IAMUserPolicy](#) or [Register-IAMGroupPolicy](#)
 - IAM API: [AttachUserPolicy](#) or [AttachGroupPolicy](#)

To change the permissions allowed by a role (AWS CLI, AWS Tools for Windows PowerShell, AWS API)

1. (Optional) To view the current permissions associated with a role, use the following commands:
 - AWS CLI: [aws iam list-role-policies](#) (to list inline policies) and [aws iam list-attached-role-policies](#) (to list managed policies)
 - AWS Tools for Windows PowerShell: [Get-IAMRolePolicies](#) (to list inline policies) and [Get-IAMAttachedRolePolicies](#) (to list managed policies)
 - IAM API: [ListRolePolicies](#) (to list inline policies) and [ListAttachedRolePolicies](#) (to list managed policies)
2. The command to update permissions for the role differs depending on whether you are updating a managed policy or an inline policy.

To update a managed policy, use one of the following commands to create a new version of the managed policy:

- AWS CLI: [aws iam create-policy-version](#)
- AWS Tools for Windows PowerShell: [New-IAMPolicyVersion](#)
- IAM API: [CreatePolicyVersion](#)

To update an inline policy, use one of the following commands:

- AWS CLI: [aws iam put-role-policy](#)
- AWS Tools for Windows PowerShell: [Write-IAMRolePolicy](#)
- IAM API: [PutRolePolicy](#)

To change the description of a role (AWS CLI, AWS API)

1. (Optional) To view the current description for a role, use the following commands:
 - AWS CLI: [aws iam get-role](#)
 - IAM API: [GetRole](#)
2. To update a role's description, use one of the following commands:
 - AWS CLI: [aws iam update-role-description](#)
 - IAM API: [UpdateRoleDescription](#)

Deleting Roles or Instance Profiles

If you no longer need a role, we recommend that you delete the role and its associated permissions. That way you don't have an unused entity that is not actively monitored or maintained.

If the role was associated with an EC2 instance, then you can also remove the role from the instance profile and then delete the instance profile.

Warning

Make sure that you do not have any Amazon EC2 instances running with the role or instance profile you are about to delete. Deleting a role or instance profile that is associated with a running instance will break any applications that are running on the instance.

Deleting a Service-Linked Role

If the role is a [service-linked role](#) (p. 136), review the documentation for the linked service to learn how to delete the role. You can view the service-linked roles in your account by going to the IAM **Roles** page

in the console. Service-linked roles appear with **(Service-linked role)** in the **Trusted entities** column of the table. A banner on the role's **Summary** page also indicates that the role is a service-linked role.

If the service does not include documentation for deleting the service-linked role, then you can use the IAM console, API, or CLI to delete the role. For more information, see [Deleting a Service-Linked Role \(p. 181\)](#).

Deleting an IAM Role (Console)

When you use the AWS Management Console to delete a role, IAM also automatically deletes the policies associated with the role. It also deletes any Amazon EC2 instance profile that contains the role.

Important

In some cases, a role might be associated with an Amazon EC2 instance profile, and the role and the instance profile might have the exact same name. In that case you can use the AWS console to delete the role and the instance profile. This linkage happens automatically for roles and instance profiles that you create them in the console. If you created the role from the AWS CLI, Tools for Windows PowerShell, or the AWS API, then the role and the instance profile might have different names. In that case you cannot use the console to delete them. Instead, you must use the AWS CLI, Tools for Windows PowerShell, or AWS API to first remove the role from the instance profile. You must then take a separate step to delete the role.

To delete a role (console)

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, choose **Roles**, and then select the check box next to the role name that you want to delete, not the name or row itself.
3. For **Role actions** at the top of the page, choose **Delete role**.
4. In the confirmation dialog box, review the service last accessed data, which shows when each of the selected roles last accessed an AWS service. This helps you to confirm whether the role is currently active. If you want to proceed, choose **Yes, Delete**. If you are sure, you can proceed with the deletion even if the service last accessed data is still loading.

Note

You cannot use the console to delete an instance profile, except when it has the exact same name as the role. In addition, you must delete the instance profile as part of the process of deleting a role as described in the preceding procedure. To delete an instance profile without also deleting the role, you must use the AWS CLI, Tools for Windows PowerShell, or AWS API. For more information, see the following sections.

Deleting an IAM Role (AWS CLI)

When you use the AWS CLI to delete a role, you must first delete the policies associated with the role. Also, if you want to delete the associated instance profile that contains the role, you must delete it separately.

To use the AWS CLI to delete a role (AWS CLI)

1. If you don't know the name of the role that you want to delete, type the following command to list the roles in your account:

```
$ aws iam list-roles
```

A list of roles with their Amazon Resource Name (ARN) is displayed. Use the role name, not the ARN, to refer to roles with the CLI commands. For example, if a role has the following ARN: `arn:aws:iam::123456789012:role/myrole`, you refer to the role as `myrole`.

2. Remove the role from all instance profiles that the role is in.
 - a. To list all instance profiles that the role is associated with, type the following command:

```
$ aws iam list-instance-profiles-for-role --role-name role-name
```

- b. To remove the role from an instance profile, type the following command for each instance profile:

```
$ aws iam remove-role-from-instance-profile --instance-profile-name instance-profile-name --role-name role-name
```

3. Delete all policies that are associated with the role.

- a. To list all policies that are in the role, type the following command:

```
$ aws iam list-role-policies --role-name role-name
```

- b. To delete each policy from the role, type the following command for each policy:

```
$ aws iam delete-role-policy --role-name role-name --policy-name policy-name
```

4. Type the following command to delete the role:

```
$ aws iam delete-role --role-name role-name
```

5. If you do not plan to reuse the instance profiles that were associated with the role, you can type the following command to delete them:

```
$ aws iam delete-instance-profile --instance-profile-name instance-profile-name
```

Deleting an IAM Role (Tools for Windows PowerShell)

When you use Windows PowerShell to delete a role, you must first delete the policies associated with the role. Also, if you want to delete the associated instance profile that contains the role, you must delete it separately.

To use the Tools for Windows PowerShell to delete a role (Tools for Windows PowerShell)

1. If you don't know the name of the role that you want to delete, type the following command to list the roles in your account:

```
PS C:\> Get-IAMRoles | Select RoleName
```

Use the role name, not the ARN, to refer to roles with the PowerShell cmdlets. For example, if a role has the following ARN: `arn:aws:iam::123456789012:role/myrole`, you refer to the role as `myrole`.

2. Remove the role from all instance profiles that the role is in. The following command gets the list of all instance profiles that contain the role, removes the role from each instance profile in the list, and then deletes the now empty instance profiles. If you plan to reuse the instance profiles, then you can omit the last cmdlet in the command.

```
PS C:\> Get-IAMInstanceProfileForRole -RoleName RoleName | Remove-IAMRoleFromInstanceProfile -RoleName RoleName | Remove-IAMInstanceProfile
```

3. Delete all policies that are associated with the role. The following command gets the list of all policies that are attached to the role and detaches each one.

```
PS C:\> Get-IAMAttachedRolePolicies -RoleName RoleName | Unregister-IAMRolePolicy -  
RoleName RoleName
```

4. Type the following command to delete the role:

```
PS C:\> Remove-IAMRole -RoleName RoleName
```

Deleting an IAM Role (AWS API)

When you use the IAM API to delete a role, you must first delete the policies associated with the role. Also, if you want to delete the associated instance profile that contains the role, you must delete it separately.

To use the AWS API to delete a role (API)

1. To list all instance profiles that a role is in, call [ListInstanceProfilesForRole](#).

To remove the role from all instance profiles that the role is in, call [RemoveRoleFromInstanceProfile](#). You must pass the role name and instance profile name.

If you are not going to reuse an instance profile that was associated with the role, you call [DeleteInstanceProfile](#) to delete it.

2. To list all policies for a role, call [ListRolePolicies](#).

To delete all policies that are associated with the role, call [DeleteRolePolicy](#). You must pass the role name and policy name.

3. Call [DeleteRole](#) to delete the role.

Related Information

For general information about instance profiles, see [Using Instance Profiles \(p. 218\)](#).

For general information about service-linked roles, see [Using Service-Linked Roles \(p. 176\)](#).

How IAM Roles Differ from Resource-based Policies

For some AWS services, you can grant cross-account access to your resources. To do this, you attach a policy directly to the resource that you want to share, instead of using a role as a proxy. The resource that you want to share must support [resource-based policies \(p. 285\)](#). Unlike a user-based policy, a resource-based policy specifies who (in the form of a list of AWS account ID numbers) can access that resource.

Cross-account access with a resource-based policy has an advantage over a role. With a resource that is accessed through a resource-based policy, the user still works in the trusted account and does not have to give up his or her user permissions in place of the role permissions. In other words, the user continues to have access to resources in the trusted account at the same time as he or she has access to the resource in the trusting account. This is useful for tasks such as copying information to or from the shared resource in the other account.

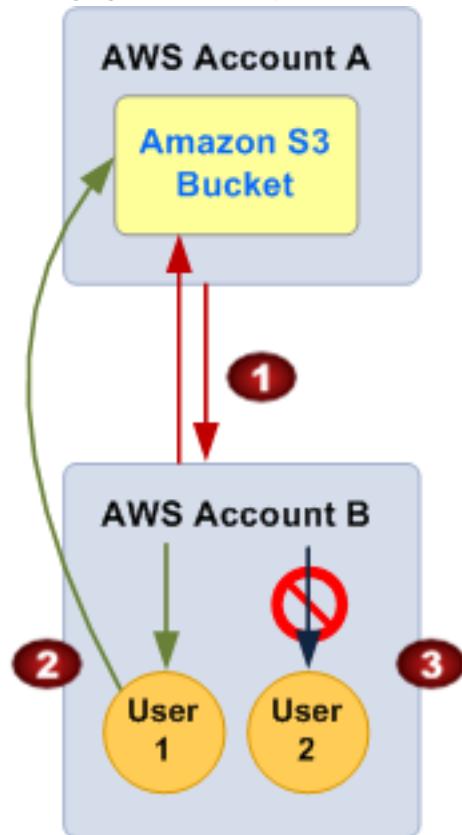
The disadvantage is that not all services support resource-based policies. A few of the AWS services that support resource-based policies are listed here:

- **Amazon S3 buckets** – The policy is attached to the bucket, but the policy controls access to both the bucket and the objects in it. For more information, go to [Access Control](#) in the *Amazon Simple Storage Service Developer Guide*.
- **Amazon Simple Notification Service (Amazon SNS) topics** – For more information, go to [Managing Access to Your Amazon SNS Topics](#) in the *Amazon Simple Notification Service Developer Guide*.
- **Amazon Simple Queue Service (Amazon SQS) queues** – For more information, go to [Appendix: The Access Policy Language](#) in the *Amazon Simple Queue Service Developer Guide*.

For a complete list of the growing number of AWS services that support attaching permission policies to resources instead of principals, see [AWS Services That Work with IAM \(p. 417\)](#) and look for the services that have **Yes** in the **Resource Based** column.

About Delegating AWS Permissions in a Resource-based Policy

After a resource grants your AWS account permissions as a principal in its resource-based policy, you can then delegate permissions to specific users or groups under your AWS account. You attach a policy to the user or group that you want to delegate the permissions to. Note that you can only delegate permissions equivalent to, or less than, the permissions granted to your account by the resource owning account. For example, if your account is granted full access to the resources of another AWS account, then you can delegate full access, list access, or any other partial access to users under your AWS account. If, on the other hand, your account is granted list access only, then you can delegate only list access. If you try to delegate more permissions than your account has, your users will still have only list access only. This is illustrated in the following figure. For information about attaching a policy to a user or group, see [Managing IAM Policies \(p. 316\)](#).



1. Account A gives account B full access to account A's S3 bucket by naming account B as a principal in the policy. As a result, account B is authorized to perform any action on account A's bucket, and the account B administrator can delegate access to its users in account B.
2. The account B administrator grants user 1 read-only access to account A's S3 bucket. User 1 can view the objects in account A's bucket. The level of access account B can delegate is equivalent to, or less than, the access the account has. In this case, the full access granted to account B is filtered to read only for user 1.
3. The account B administrator does not give access to user 2. Because users by default do not have any permissions except those that are explicitly granted, user 2 does not have access to account A's Amazon S3 bucket.

Important

In the preceding example, if account B had used wildcards (*) to give user 1 full access to all its resources, user 1 would automatically have access to any resources that account B has access to, including access granted by other accounts to those accounts' resources. In this case, user 1 would have access to any Account A resources granted to account B, in addition to those explicitly granted to user 1.

IAM evaluates a user's permissions at the time the user makes a request. Therefore, if you use wildcards (*) to give users full access to your resources, users are able to access any resources that your AWS account has access to, even resources you add or gain access to after creating the user's policy.

For information about permissions, policies, and the permission policy language that you use to write policies, see [Access Management \(p. 274\)](#).

Important

Give access only to entities you trust, and give the minimum amount of access necessary.

Whenever the trusted entity is another AWS account, that account can in turn delegate access to any of its IAM users. The trusted AWS account can delegate access only to the extent that it has been granted access; it cannot delegate more access than the account itself has been granted.

Temporary Security Credentials

You can use the AWS Security Token Service (AWS STS) to create and provide trusted users with temporary security credentials that can control access to your AWS resources. Temporary security credentials work almost identically to the long-term access key credentials that your IAM users can use, with the following differences:

- Temporary security credentials are *short-term*, as the name implies. They can be configured to last for anywhere from a few minutes to several hours. After the credentials expire, AWS no longer recognizes them or allows any kind of access from API requests made with them.
- Temporary security credentials are not stored with the user but are generated dynamically and provided to the user when requested. When (or even before) the temporary security credentials expire, the user can request new credentials, as long as the user requesting them still has permissions to do so.

These differences lead to the following advantages for using temporary credentials:

- You do not have to distribute or embed long-term AWS security credentials with an application.
- You can provide access to your AWS resources to users without having to define an AWS identity for them. Temporary credentials are the basis for [roles and identity federation \(p. 135\)](#).
- The temporary security credentials have a limited lifetime, so you do not have to rotate them or explicitly revoke them when they're no longer needed. After temporary security credentials expire, they cannot be reused. You can specify how long the credentials are valid, up to a maximum limit.

AWS STS and AWS Regions

Temporary security credentials are generated by AWS STS. By default, AWS STS is a global service with a single endpoint at <https://sts.amazonaws.com>. However, you can also choose to make AWS STS API calls to endpoints in any other supported region. This can reduce latency (server lag) by sending the requests to servers in a region that is geographically closer to you. No matter which region your credentials come from, they work globally. For more information, see [Activating and Deactivating AWS STS in an AWS Region \(p. 257\)](#).

Common Scenarios for Temporary Credentials

Temporary credentials are useful in scenarios that involve identity federation, delegation, cross-account access, and IAM roles.

Identity Federation

You can manage your user identities in an external system outside of AWS and grant users who sign in from those systems access to perform AWS tasks and access your AWS resources. IAM supports two types of identity federation. In both cases, the identities are stored outside of AWS. The distinction is where the external system resides—in your data center or an external third party on the web. For more information about external identity providers, see [Identity Providers and Federation \(p. 143\)](#).

- **Enterprise identity federation** – You can authenticate users in your organization's network, and then provide those users access to AWS without creating new AWS identities for them and requiring them to sign in with a separate user name and password. This is known as the *single sign-on* (SSO) approach to temporary access. AWS STS supports open standards like Security Assertion Markup Language (SAML) 2.0, with which you can use Microsoft AD FS to leverage your Microsoft Active Directory. You can also use SAML 2.0 to manage your own solution for federating user identities. For more information, see [About SAML 2.0-based Federation \(p. 149\)](#).
- **Custom federation broker** – You can use your organization's authentication system to grant access to AWS resources. For an example scenario, see [Creating a URL that Enables Federated Users to Access the AWS Management Console \(Custom Federation Broker\) \(p. 169\)](#).
- **Federation using SAML 2.0** – You can use your organization's authentication system and SAML to grant access to AWS resources. For more information and an example scenario, see [About SAML 2.0-based Federation \(p. 149\)](#).
- **Web identity federation** – You can let users sign in using a well-known third party identity provider such as Login with Amazon, Facebook, Google, or any OpenID Connect (OIDC) 2.0 compatible provider. You can exchange the credentials from that provider for temporary permissions to use resources in your AWS account. This is known as the *web identity federation* approach to temporary access. When you use web identity federation for your mobile or web application, you don't need to create custom sign-in code or manage your own user identities. Using web identity federation helps you keep your AWS account secure, because you don't have to distribute long-term security credentials, such as IAM user access keys, with your application. For more information, see [About Web Identity Federation \(p. 143\)](#).

AWS STS web identity federation supports Login with Amazon, Facebook, Google, and any OpenID Connect (OIDC)-compatible identity provider.

Note

For mobile applications, we recommend that you use Amazon Cognito. You can use this service with the [AWS Mobile SDK for iOS](#) and the [AWS Mobile SDK for Android and Fire OS](#) to create unique identities for users and authenticate them for secure access to your AWS resources. Amazon Cognito supports the same identity providers as AWS STS, and also supports unauthenticated (guest) access and lets you migrate user data when a user signs in. Amazon Cognito also provides APIs for synchronizing user data so that it is preserved as users move between devices. For more information, see the following:

- [Amazon Cognito Identity in the AWS Mobile SDK for iOS Developer Guide](#)
- [Amazon Cognito Identity in the AWS Mobile SDK for Android Developer Guide](#)

Roles for Cross-account Access

Many organizations maintain more than one AWS account. Using roles and cross-account access, you can define user identities in one account, and use those identities to access AWS resources in other accounts that belong to your organization. This is known as the *delegation* approach to temporary access. For more information, see [Creating a Role to Delegate Permissions to an IAM User \(p. 184\)](#).

Roles for Amazon EC2

If you run applications on Amazon EC2 instances and those applications need access to AWS resources, you can provide temporary security credentials to your instances when you launch them. These temporary security credentials are available to all applications that run on the instance, so you don't need to store any long-term credentials on the instance. For more information, see [Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances \(p. 215\)](#).

Other AWS Services

You can use temporary security credentials to access most AWS services. For a list of the services that accept temporary security credentials, see [AWS Services That Work with IAM \(p. 417\)](#).

Requesting Temporary Security Credentials

To request temporary security credentials, you can use the AWS STS API actions. You can use AWS Security Token Service (AWS STS) to create and provide trusted users with temporary security credentials that can control access to your AWS resources. For more information about AWS STS, see [Temporary Security Credentials \(p. 231\)](#).

To call the APIs, you can use one of the [AWS SDKs](#), which are available for a variety of programming languages and environments, including Java, .NET, Python, Ruby, Android, and iOS. The SDKs take care of tasks such as cryptographically signing your requests, retrying requests if necessary, and handling error responses. You can also use the AWS STS Query API, which is described in the [AWS Security Token Service API Reference](#). Finally, two command line tools support the AWS STS commands: the [AWS Command Line Interface](#), and the [AWS Tools for Windows PowerShell](#).

The AWS STS API actions return temporary security credentials that consist of an access key and a session token. The access key consists of an access key ID and a secret key. Users (or an application that the user runs) can use these credentials to access your resources. When the credentials are created, they are associated with an IAM access control policy that limits what the user can do when using the credentials. For more information, see [Using Temporary Security Credentials to Request Access to AWS Resources \(p. 242\)](#).

Important

Although temporary security credentials are short lived, users who have temporary access can make lasting changes to your AWS resources. For example, if a user with temporary access launches an Amazon EC2 instance, the instance can continue to run and incur charges to your AWS account even after the user's temporary security credentials expire.

Note

The size of the security token that STS APIs return is not fixed. We strongly recommend that you make no assumptions about the maximum size. As of this writing, the typical size is less than 4096 bytes, but that can vary. Also, future updates to AWS might require larger sizes.

Using AWS STS with AWS Regions

You can send AWS STS API calls either to a global endpoint or to one of the regional endpoints. If you choose an endpoint closer to you, you can reduce latency and improve the performance of your API calls. You also can choose to direct your calls to an alternative regional endpoint if you can no longer communicate with the original endpoint. If you are using one of the various AWS SDKs, then use that SDK's method to select a region before you make the API call. If you are manually constructing HTTP API requests, then you must direct the request to the correct endpoint yourself. For more information, see the [AWS STS section of Regions and Endpoints](#) and [Activating and Deactivating AWS STS in an AWS Region \(p. 257\)](#).

Following are the APIs that you can use to acquire temporary credentials for use in your AWS environment and applications.

AssumeRole—Cross-Account Delegation and Federation Through a Custom Identity Broker

This API action is useful for allowing existing IAM users to access AWS resources that they don't already have access to, such as resources in another AWS account. It is also useful for existing IAM users as a means to temporarily gain privileged access—for example, to provide multi-factor authentication (MFA). You must call this API using existing IAM user credentials. For more information, see [Creating a Role to Delegate Permissions to an IAM User \(p. 184\)](#) and [Configuring MFA-Protected API Access \(p. 107\)](#).

This call must be made using valid AWS security credentials. When you make this call, you pass the following information:

- The Amazon Resource Name (ARN) of the role that the app should assume.
- The duration, which specifies how long the temporary security credentials are valid. The minimum is 15 minutes (900 seconds) and the maximum (and the default) is 1 hour (3600 seconds). You need to pass this value only if you want the temporary credentials to expire before 1 hour. This is separate from the duration of a console session that you might request using these credentials. The request to the federation endpoint for a console sign-in token takes a `SessionDuration` parameter that specifies the maximum length of the console session, separately from the `DurationSeconds` parameter on this API. For more information, see [Creating a URL that Enables Federated Users to Access the AWS Management Console \(Custom Federation Broker\) \(p. 169\)](#).
- A role session name, which is a string value that you can use to identify the session. This value can be captured and logged by CloudTrail to help you distinguish between your role users during an audit.
- Optionally, a policy (in JSON format). This policy is combined with the policy associated with the role. If specified, the permissions are the intersection of those granted to the role and those granted by this policy. You can use this policy to further restrict the access permissions that are associated with the temporary credentials, beyond the restrictions already established by the role permission policy. Note that this policy cannot be used to elevate privileges beyond what the assumed role is allowed to access.
- If configured to use multi-factor authentication (MFA), then you include the identifier for an MFA device and the one-time code provided by that device.
- An optional `ExternalID` value that can be used when delegating access to your account to a third-party. This value helps ensure that only the specified third-party can access the role. For more information, see [How to Use an External ID When Granting Access to Your AWS Resources to a Third Party \(p. 187\)](#).

The following example shows a sample request and response using `AssumeRole`. In this example, the request includes the name for the session named Bob. The `Policy` parameter includes a JSON document that specifies that the resulting credentials have permissions to access only Amazon S3.

Example Request

```
https://sts.amazonaws.com/  
?Version=2011-06-15  
&Action=AssumeRole  
&RoleSessionName=Bob  
&RoleArn=arn:aws::iam::123456789012:role/demo  
&Policy=%7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A%5B%7B%22Sid%22%3A  
%20%22Stmt1%22%2C%22Effect%22%3A%20%22Allow%22%2C%22Action%22%3A%20%22s3%3A*%22%2C  
%22Resource%22%3A%20%22*%22%7D%5D%7D  
&DurationSeconds=3600  
&ExternalId=123ABC  
&AUTHPARAMS
```

Note

The policy value shown in the example above is the URL-encoded version of the following policy:

```
{"Version": "2012-10-17", "Statement":  
[{"Sid": "Stmt1", "Effect": "Allow", "Action": "s3:*", "Resource": "*"}]}
```

Also, note that the AUTHPARAMS parameter in the preceding example is meant as a placeholder for the authentication information—that is, the *signature*—that you must include with AWS HTTP API requests. We recommend using the [AWS SDKs](#) to create API requests, and one benefit of doing so is that the SDKs handle request signing for you. If you must create and sign API requests manually, go to [Signing AWS Requests By Using Signature Version 4](#) in the *Amazon Web Services General Reference* to learn how to sign a request.

This API supports a parameter for DurationSeconds that specifies how long the temporary credentials are valid. This is not the same as the duration of a console session that might request using those temporary credentials. You can request a console sign-in token by calling the federation endpoint and supplying the temporary credentials to get a sign-in token for the console. That console request uses a different SessionDuration parameter of up to 12 hours. For more information, see [Creating a URL that Enables Federated Users to Access the AWS Management Console \(Custom Federation Broker\) \(p. 169\)](#).

In addition to the temporary security credentials, the response includes the Amazon Resource Name (ARN) for the federated user and the expiration time of the credentials.

Example Response

```
<AssumeRoleResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">  
<AssumeRoleResult>  
<Credentials>  
<SessionToken>  
AQoDYXdzEPT//////////wEXAMPLEtc764bNrC9SAPBSM22wD0k4x4HIZ8j4FZTwQW  
LWsKWHGBuFqwAeMicRXmxfpSPfleoIYRqTflfKD8YUuwtAx7mSEI/qkPpKPi/kMcGd  
OrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDyOKPkYQDYwT7WZ0wg5VSXDvp75YU  
9HFv1Rd8Tx6q6fE8YQcHNVXAkiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL641IZbqBAz  
+scqKmlzm8FDdrypNC9Yjc8fPOLn9FX9KSvKTr4rvx3iSi1TJabiQwj2ICCR/oLxBA==  
</SessionToken>  
<SecretAccessKey>  
wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY  
</SecretAccessKey>  
<Expiration>2011-07-15T23:28:33.359Z</Expiration>  
<AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>  
</Credentials>  
<AssumedRoleUser>  
<Arn>arn:aws:sts::123456789012:assumed-role/demo/Bob</Arn>  
<AssumedRoleId>ARO123EXAMPLE123:Bob</AssumedRoleId>  
</AssumedRoleUser>  
<PackedPolicySize>6</PackedPolicySize>
```

```
</AssumeRoleResult>
<ResponseMetadata>
<RequestId>c6104cbe-af31-11e0-8154-cbc7ccf896c7</RequestId>
</ResponseMetadata>
</AssumeRoleResponse>
```

Note

AssumeRole stores the policy in a packed format. AssumeRole returns the size as a percentage of the maximum size allowed so you can adjust the calling parameters. For more information about the size constraints on the policy, go to [AssumeRole](#) in the *AWS Security Token Service API Reference*.

AssumeRoleWithWebIdentity—Federation Through a Web-based Identity Provider

This API returns a set of temporary security credentials for federated users who are authenticated through a public identity provider such as Login with Amazon, Facebook, Google, or any OpenID Connect (OIDC)-compatible identity provider. This API is useful for creating mobile applications or client-based web applications that require access to AWS in which users do not have their own AWS or IAM identities. For more information, see [About Web Identity Federation \(p. 143\)](#).

Note

Instead of directly calling `AssumeRoleWithWebIdentity`, we recommend that you use Amazon Cognito and the Amazon Cognito credentials provider with the AWS SDKs for mobile development. For more information, see the following:

- [Amazon Cognito Identity in the AWS Mobile SDK for Android Developer Guide](#)
- [Amazon Cognito Identity in the AWS Mobile SDK for iOS Developer Guide](#)

If you are not using Amazon Cognito, you call the `AssumeRoleWithWebIdentity` action of AWS STS. This is an unsigned call, meaning that the app does not need to have access to any AWS security credentials in order to make the call. When you make this call, you pass the following information:

- The Amazon Resource Name (ARN) of the role that the app should assume. If your app supports multiple ways for users to sign in, you must define multiple roles, one per identity provider. The call to `AssumeRoleWithWebIdentity` should include the ARN of the role that is specific to the provider through which the user signed in.
- The token that the app gets from the IdP after the app authenticates the user.
- The duration, which specifies how long the temporary security credentials are valid. The minimum is 15 minutes (900 seconds) and the maximum (and the default) is 1 hour (3600 seconds). You need to pass this value only if you want the temporary credentials to expire before 1 hour. This is separate from the duration of a console session that you might request using these credentials. The request to the federation endpoint for a console sign-in token takes a `SessionDuration` parameter that specifies the maximum length of the console session, separately from the `DurationSeconds` parameter on this API. For more information, see [Creating a URL that Enables Federated Users to Access the AWS Management Console \(Custom Federation Broker\) \(p. 169\)](#).
- A role session name, which is a string value that you can use to identify the session. This value can be captured and logged by CloudTrail to help you distinguish between your role users during an audit.
- Optionally, a policy (in JSON format). This policy is combined with the policy associated with the role. If specified, the permissions are the intersection of those granted to the role and those granted by this policy. You can use this policy to further restrict the access permissions that are associated with the temporary credentials, beyond the restrictions already established by the role permission policy. Note that this policy cannot be used to elevate privileges beyond what the assumed role is allowed to access.

Note

Because a call to `AssumeRoleWithWebIdentity` is not signed (encrypted), you should only include this optional policy if the request is not being transmitted through an untrusted intermediary who could alter the policy to remove the restrictions.

When you call `AssumeRoleWithWebIdentity`, AWS verifies the authenticity of the token. For example, depending on the provider, AWS might make a call to the provider and include the token that the app has passed. Assuming that the identity provider validates the token, AWS returns the following information to you:

- A set of temporary security credentials. These consist of an access key ID, a secret access key, and a session token.
- The role ID and the ARN of the assumed role.
- A `SubjectFromWebIdentityToken` value that contains the unique user ID.

When you have the temporary security credentials, you can use them to make AWS API calls. This is the same process as making an AWS API call with long-term security credentials, except that you must include the session token, which lets AWS verify that the temporary security credentials are valid.

Your app should cache the credentials. As noted, by default the credentials expire after an hour. If you are not using the [AmazonSTSCredentialsProvider](#) action in the AWS SDK, it's up to you and your app to call `AssumeRoleWithWebIdentity` again to get a new set of temporary security credentials before the old ones expire.

AssumeRoleWithSAML—Federation Through an Enterprise Identity Provider Compatible with SAML 2.0

This API returns a set of temporary security credentials for federated users who are authenticated by your organization's existing identity system and who use [SAML 2.0](#) (Security Assertion Markup Language) to pass authentication and authorization information to AWS. This API is useful in organizations that have integrated their identity systems (such as Windows Active Directory or OpenLDAP) with software that can produce SAML assertions to provide information about user identity and permissions (such as Active Directory Federation Services or Shibboleth). For more information, see [About SAML 2.0-based Federation \(p. 149\)](#).

This is an unsigned call, which means that the app does not need to have access to any AWS security credentials in order to make the call. When you make this call, you pass the following information:

- The Amazon Resource Name (ARN) of the role that the app should assume.
- The ARN of the SAML provider created in IAM that describes the identity provider.
- The SAML assertion, encoded in base-64, that was provided by the SAML identity provider in its authentication response to the sign-in request from your app.
- The duration, which specifies how long the temporary security credentials are valid. The maximum (and the default) is 1 hour (3600 seconds). You need to pass this value only if you want the temporary credentials to expire before 1 hour. The minimum duration for the credentials is 15 minutes (900 seconds). This is separate from the duration of a console session that you might request using these credentials. The request to the federation endpoint for a console sign-in token takes a `SessionDuration` parameter that specifies the maximum length of the console session, separately from the `DurationSeconds` parameter on this API. For more information, see [Creating a URL that Enables Federated Users to Access the AWS Management Console \(Custom Federation Broker\) \(p. 169\)](#).
- A policy (in JSON format). This policy is combined with the policy associated with the role. If specified, the permissions are the intersection of those granted to the role and those granted by this policy. You can use this policy to further restrict the access permissions that are associated with the temporary

credentials, beyond the restrictions already established by the role permission policy. Note that this policy cannot be used to elevate privileges beyond what the assumed role is allowed to access.

When you call `AssumeRoleWithSAML`, AWS verifies the authenticity of the SAML assertion. Assuming that the identity provider validates the assertion, AWS returns the following information to you:

- A set of temporary security credentials. These consist of an access key ID, a secret access key, and a session token.
- The role ID and the ARN of the assumed role.
- An `Audience` value that contains the value of the `Recipient` attribute of the `SubjectConfirmationData` element of the SAML assertion.
- An `Issuer` value that contains the value of the `Issuer` element of the SAML assertion.
- A `NameQualifier` element that contains a hash value built from the `Issuer` value, the AWS account ID, and the friendly name of the SAML provider. When combined with the `Subject` element, they can uniquely identify the federated user.
- A `Subject` element that contains the value of the `NameID` element in the `Subject` element of the SAML assertion.
- A `SubjectType` element that indicates the format of the `Subject` element. The value can be `persistent`, `transient`, or the full `Format` URI from the `Subject` and `NameID` elements used in your SAML assertion. For information about the `NameID` element's `Format` attribute, see [Configuring SAML Assertions for the Authentication Response \(p. 163\)](#).

When you have the temporary security credentials, you can use them to make AWS API calls. This is the same process as making an AWS API call with long-term security credentials, except that you must include the session token, which lets AWS verify that the temporary security credentials are valid.

Your app should cache the credentials. By default the credentials expire after an hour. If you are not using the `AmazonSTSCredentialsProvider` action in the AWS SDK, it's up to you and your app to call `AssumeRoleWithSAML` again to get a new set of temporary security credentials before the old ones expire.

GetFederationToken—Federation Through a Custom Identity Broker

This API returns a set of temporary security credentials for federated users. This API differs from `AssumeRole` in that the default expiration period is substantially longer (up to 36 hours instead of up to 1 hour). The longer expiration period can help reduce the number of calls to AWS because you do not need to get new credentials as often. For more information, see [Requesting Temporary Security Credentials \(p. 233\)](#).

The `GetFederationToken` call returns temporary security credentials that consist of the security token, access key, secret key, and expiration. You can use `GetFederationToken` if you want to manage permissions inside your organization (for example, using the proxy application to assign permissions). To view a sample application that uses `GetFederationToken`, go to [Identity Federation Sample Application for an Active Directory Use Case](#) in the *AWS Sample Code & Libraries*.

The following example shows a sample request and response that uses `GetFederationToken`. In this example, the request includes the name for a federated user named Jean. The `Policy` parameter includes a JSON document that specifies that the resulting credentials have permissions to access only Amazon S3. In addition to the temporary security credentials, the response includes the Amazon Resource Name (ARN) for the federated user and the expiration time of the credentials.

Example Request

```
https://sts.amazonaws.com/
```

```
?Version=2011-06-15
&Action=GetFederationToken
&Name=Jean
&Policy=%7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A%5B%7B%22Sid%22%3A
%22Stmt1%22%2C%22Effect%22%3A%22Allow%22%2C%22Action%22%3A%22s3%3A*%22%2C%22Resource%22%3A
%22*%22%7D%5D%7D
&DurationSeconds=3600
&AUTHPARAMS
```

Note

The policy value shown in the example above is the URL-encoded version of this policy:

```
{"Version": "2012-10-17", "Statement": [
    {"Sid": "Stmt1", "Effect": "Allow", "Action": "s3:*", "Resource": "*"}]}
```

Also, note that the `&AUTHPARAMS` parameter in the preceding example is meant as a placeholder for the authentication information—that is, the *signature*—that you must include with AWS HTTP API requests. We recommend using the [AWS SDKs](#) to create API requests, and one benefit of doing so is that the SDKs handle request signing for you. If you must create and sign API requests manually, go to [Signing AWS Requests By Using Signature Version 4](#) in the *Amazon Web Services General Reference* to learn how to sign a request.

Example Response

```
<GetFederationTokenResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
<GetFederationTokenResult>
<Credentials>
  <SessionToken>
    AQoDYXdzEPT//////////wEXAMPLEtc764bNrC9SAPBSM22wD0k4x4HIZ8j4FZTwdQW
    LWsKWHGBuFqwAeMicRXmxfpSPfIeoIYRqTflfKD8YUuwthAx7mSEI/qkPpKPi/kMcGd
    QrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDy0KPkyQDYwT7WZ0wq5VSXDvp75YU
    9HFv1Rd8Tx6q6fE8YQcHNVXAkiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL64lIZbqBAz
    +scqKmlzm8FDrypNC9Yjc8fPOLn9FX9KSvKTr4rvx3iS1lTJabiQwj2ICCEAMPLE==
  </SessionToken>
  <SecretAccessKey>
    wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPEKEY
  </SecretAccessKey>
  <Expiration>2011-07-15T23:28:33.359Z</Expiration>
  <AccessKeyId>AKIAIOSFODNN7EXAMPLE;</AccessKeyId>
</Credentials>
<FederatedUser>
  <Arn>arn:aws:sts::123456789012:federated-user/Jean</Arn>
  <FederatedUserId>123456789012:Jean</FederatedUserId>
</FederatedUser>
<PackedPolicySize>6</PackedPolicySize>
</GetFederationTokenResult>
<ResponseMetadata>
  <RequestId>c6104cbe-af31-11e0-8154-cbc7ccf896c7</RequestId>
</ResponseMetadata>
</GetFederationTokenResponse>
```

Note

`GetFederationToken` stores the policy in a packed format. The action returns the size as a percentage of the maximum size allowed so that you can adjust the calling parameters. For more information about size constraints on the policy, go to [GetFederationToken](#) in the *AWS Security Token Service API Reference*.

If you prefer to grant permissions at the resource level (for example, you attach a policy to an Amazon S3 bucket), you can omit the `Policy` parameter. However, if you do not include a policy for the federated user, the temporary security credentials will not grant any permissions. In this case, you *must* use resource policies to grant the federated user access to your AWS resources.

For example, if your AWS account number is 111122223333, and you have an Amazon S3 bucket that you want to allow Susan to access even though her temporary security credentials don't include a policy

for the bucket, you would need to ensure that the bucket has a policy with an ARN that matches Susan's ARN, such as `arn:aws:sts::111122223333:federated-user/Susan`.

GetSessionToken—Temporary Credentials for Users in Untrusted Environments

This API returns a set of temporary security credentials to an existing IAM user. It is useful for providing enhanced security, for example, to restrict AWS requests to only when MFA is enabled for the IAM user. Because the credentials are temporary, they provide enhanced security when you have an IAM user who accesses your resources through a less secure environment, such as a mobile device or web browser. For more information, see [Requesting Temporary Security Credentials \(p. 233\)](#) or [GetSessionToken](#) in the [AWS Security Token Service API Reference](#).

By default, temporary security credentials for an IAM user are valid for a maximum of 12 hours, but you can request a duration as short as 15 minutes or as long as 36 hours. For security reasons, a token for an AWS account root user is restricted to a duration of one hour.

`GetSessionToken` returns temporary security credentials consisting of a security token, an access key ID, and a secret access key. The following example shows a sample request and response using `GetSessionToken`. The response also includes the expiration time of the temporary security credentials.

Example Request

```
https://sts.amazonaws.com/  
?Version=2011-06-15  
&Action=GetSessionToken  
&DurationSeconds=3600  
&AUTHPARAMS
```

Note

The `&AUTHPARAMS` parameter in the preceding example is meant as a placeholder for the authentication information—that is, the *signature*—that you must include with AWS HTTP API requests. We recommend using the [AWS SDKs](#) to create API requests, and one benefit of doing so is that the SDKs handle request signing for you. If you must create and sign API requests manually, go to [Signing AWS Requests By Using Signature Version 4](#) in the *Amazon Web Services General Reference* to learn how to sign a request.

Example Response

```
<GetSessionTokenResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">  
<GetSessionTokenResult>  
<Credentials>  
<SessionToken>  
AQoEXAMPLE4aoAH0gNCAPyJxz4B1CFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrrH3c/L  
To6UDdyJwOOvEVpvLXCrrrUtdnniCEXAMPLE/IvU1dYUg2RVAJBanLiHb4IgRmpRV3z  
rkuWJOgs8IZZaIv2BXIa2R4OljkBN9bkUDNCJiBeb/AXlzBBko7b15fjrBs2+cTQtp  
Z3CYWFxG8C5zqx37wnOE49mRl/+OtkIKGO7fAE  
</SessionToken>  
<SecretAccessKey>  
wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY  
</SecretAccessKey>  
<Expiration>2011-07-11T19:55:29.611Z</Expiration>  
<AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>  
</Credentials>  
</GetSessionTokenResult>  
<ResponseMetadata>  
<RequestId>58c5dbae-abef-11e0-8cfe-09039844ac7d</RequestId>  
</ResponseMetadata>  
</GetSessionTokenResponse>
```

Optionally, the `GetSessionToken` request can include `SerialNumber` and `TokenCode` values for AWS multi-factor authentication (MFA) verification. If the provided values are valid, AWS STS provides temporary security credentials that include the state of MFA authentication so that the temporary security credentials can be used to access the MFA-protected API actions or AWS websites for as long as the MFA authentication is valid.

The following example shows a `GetSessionToken` request that includes an MFA verification code and device serial number.

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=GetSessionToken
&DurationSeconds=7200
&SerialNumber=YourMFADeviceSerialNumber
&TokenCode=123456
&AUTHPARAMS
```

Note

The call to AWS STS can be to the global endpoint or to any of the regional endpoints for which you activate your AWS account. For more information, see the [AWS STS section of Regions and Endpoints](#).

Also, note that the `&AUTHPARAMS` parameter in the preceding example is meant as a placeholder for the authentication information—that is, the *signature*—that you must include with AWS HTTP API requests. We recommend using the [AWS SDKs](#) to create API requests, and one benefit of doing so is that the SDKs handle request signing for you. If you must create and sign API requests manually, go to [Signing AWS Requests By Using Signature Version 4](#) in the *Amazon Web Services General Reference* to learn how to sign a request.

Comparing the AWS STS APIs

The following table compares features of the actions (APIs) in AWS STS that return temporary security credentials.

Comparing your API options

AWS STS API	Who can call	Credential lifetime (min/max/default)	MFA support*	Passed policy support*	Restrictions on resulting temporary credentials
<code>AssumeRole</code>	IAM user or user with existing temporary security credentials	15m/1hr/1d	Yes	Yes	Cannot call <code>GetFederationToken</code> or <code>GetSessionToken</code> .
<code>AssumeRoleWithSAML</code>	Any user caller must pass a SAML authentication response that indicates authentication from a known identity provider	15m/1hr/1d	No	Yes	Cannot call <code>GetFederationToken</code> or <code>GetSessionToken</code> .
<code>AssumeRoleWithWebIdentity</code>	Any user caller must pass a web identity token that indicates authentication from a known identity provider	15m/1hr/1d	No	Yes	Cannot call <code>GetFederationToken</code> or <code>GetSessionToken</code> .

AWS STS API	Who can call	Credential lifetime (min/max/default)	MFA support*	Passed policy support*	Restrictions on resulting temporary credentials
	a known identity provider				
GetFederationToken	IAM user or AWS account root user	IAM user: 15m/36hr/12hr Root user: 15m/1hr/1hr	No	Yes	Cannot call IAM APIs directly. SSO to console is allowed.* Cannot call AWS STS APIs except GetCallerIdentity .
GetSessionToken	IAM user or root user	IAM user: 15m/36hr/12hr Root user: 15m/1hr/1hr	Yes	No	Cannot call IAM APIs unless MFA information is included with the request. Cannot call AWS STS APIs except AssumeRole or GetCallerIdentity . Single sign-on (SSO) to console is not allowed, but any user with a password (root or IAM user) can sign into the console.*

- **MFA support.** You can include information about a multi-factor authentication (MFA) device when you call the [AssumeRole](#) and [GetSessionToken](#) APIs. This ensures that the temporary security credentials that result from the API call can be used only by users who are authenticated with an MFA device. For more information, see [Configuring MFA-Protected API Access \(p. 107\)](#).
- **Passed policy support.** You can pass an IAM policy as a parameter to most of the AWS STS APIs to be used in conjunction with other policies affecting the user (if any) to determine what the user is allowed to do with the temporary credentials that result from the API call. For more information, see the following topics:
 - [Permissions for AssumeRole, AssumeRoleWithSAML, and AssumeRoleWithWebIdentity \(p. 246\)](#)
 - [Permissions for GetFederationToken \(p. 249\)](#)
 - [Permissions for GetSessionToken \(p. 252\)](#)
- **Single sign-on (SSO) to the console.** To support SSO, AWS lets you call a federation endpoint (<https://signin.aws.amazon.com/federation>) and pass temporary security credentials. The endpoint returns a token that you can use to construct a URL that signs a user directly into the console without requiring a password. For more information, see [Enabling SAML 2.0 Federated Users to Access the AWS Management Console \(p. 167\)](#) and [How to Enable Cross-Account Access to the AWS Management Console](#) in the AWS Security Blog.

Using Temporary Security Credentials to Request Access to AWS Resources

You can use temporary security credentials to make programmatic requests for AWS resources with the [AWS SDKs](#) or API calls, the same way that you can use long-term security credentials such as IAM user credentials. However, there are a few differences:

- When you make a call using temporary security credentials, the call must include a session token, which is returned along with those temporary credentials. AWS uses the session token to validate the temporary security credentials.
- The temporary credentials expire after a specified interval. After the credentials expire, any calls that you make with those credentials will fail, so you must get a new set of credentials.

If you are using the [AWS SDKs](#), the [AWS Command Line Interface](#) (AWS CLI), or the [Tools for Windows PowerShell](#), the way in which you get and use temporary security credentials differs depending on the context. If you are running code, AWS CLI, or Tools for Windows PowerShell commands inside an EC2 instance, you can take advantage of roles for Amazon EC2. Otherwise, you can call an AWS STS API to get the temporary credentials, and then use them explicitly to make calls to AWS services.

Note

You can use AWS Security Token Service (AWS STS) to create and provide trusted users with temporary security credentials that can control access to your AWS resources. For more information about AWS STS, see [Temporary Security Credentials \(p. 231\)](#). AWS STS is a global service that has a default endpoint at <https://sts.amazonaws.com>. This endpoint is in the US East (Ohio) region, although credentials that you get from this and other endpoints are valid globally and work with services and resources in any region. You can also choose to make AWS STS API calls to endpoints in any of the supported regions. This can reduce latency by making the requests from servers in a region that is geographically closer to you. No matter which region your credentials come from, they work globally. For more information, see [Activating and Deactivating AWS STS in an AWS Region \(p. 257\)](#).

Contents

- [Using Temporary Credentials in Amazon EC2 Instances \(p. 243\)](#)
- [Using Temporary Security Credentials with the AWS SDKs \(p. 244\)](#)
- [Using Temporary Security Credentials with the AWS CLI \(p. 244\)](#)
- [Using Temporary Security Credentials with the Tools for Windows PowerShell \(p. 245\)](#)
- [Using Temporary Security Credentials with APIs \(p. 246\)](#)
- [More Information \(p. 246\)](#)

Using Temporary Credentials in Amazon EC2 Instances

If you want to run AWS CLI commands or code inside an EC2 instance, the recommended way to get credentials is to use [roles for Amazon EC2](#). You create an IAM role that specifies the permissions that you want to grant to applications that run on the EC2 instances. When you launch the instance, you associate the role with the instance.

Applications, AWS CLI, and Tools for Windows PowerShell commands that run on the instance can then get automatic temporary security credentials from the instance metadata. You do not have to explicitly get the temporary security credentials—the AWS SDKs, AWS CLI, and Tools for Windows PowerShell automatically get the credentials from the EC2 instance metadata service and use them. The temporary credentials have the permissions that you define for the role that is associated with the instance.

For more information and for examples, see the following:

- [Using IAM Roles to Grant Access to AWS Resources on Amazon Elastic Compute Cloud](#) — AWS SDK for Java
- [Granting Access Using an IAM Role](#) — AWS SDK for .NET
- [Creating a Role](#) — AWS SDK for Ruby

Using Temporary Security Credentials with the AWS SDKs

To use temporary security credentials in code, you programmatically call an AWS STS API like `AssumeRole`, extract the resulting credentials and session token, and then use those values as credentials for subsequent calls to AWS. The following example shows pseudocode for how to use temporary security credentials if you're using an AWS SDK:

```
assumeRoleResult = AssumeRole(role-arn);
tempCredentials = new SessionAWSCredentials(
    assumeRoleResult.AccessKeyId,
    assumeRoleResult.SecretAccessKey,
    assumeRoleResult.SessionToken);
s3Request = CreateAmazonS3Client(tempCredentials);
```

For an example written in Python (using the [AWS SDK for Python \(Boto\)](#)) that shows how to call `AssumeRole` to get temporary security credentials, and then use those credentials to make a call to Amazon S3, see [Switching to an IAM Role \(API\) \(p. 214\)](#).

For details about how to call `AssumeRole`, `GetFederationToken`, and other APIs, and about how to get the temporary security credentials and session token from the result, see the documentation for the SDK that you're working with. You can find the documentation for all the AWS SDKs on the main [AWS documentation page](#).

You must make sure that you get a new set of credentials before the old ones expire. In some SDKs, you can use a provider that manages the process of refreshing credentials for you; check the documentation for the SDK you're using.

Using Temporary Security Credentials with the AWS CLI

You can use temporary security credentials with the AWS CLI. This can be useful for testing policies.

Using the AWS CLI, you can call an AWS STS API like `AssumeRole` or `GetFederationToken` and then capture the resulting output. The following example shows a call to `AssumeRole` that sends the output to a file. In the example, the `profile` parameter is assumed to be a profile in the AWS CLI configuration file and is assumed to reference credentials for an IAM user who has permissions to assume the role.

```
$ aws sts assume-role --role-arn arn:aws:iam::123456789012:role/role-name --role-session-name "RoleSession1" --profile IAM-user-name > assume-role-output.txt
```

When the command is finished, you can extract the access key ID, secret access key, and session token from wherever you've routed it, either manually or by using a script. You can then assign these values to environment variables.

When you run AWS CLI commands, the AWS CLI looks for credentials in a specific order—first in environment variables and then in the configuration file. Therefore, after you've put the temporary credentials into environment variables, the AWS CLI uses those credentials by default. (If you specify a `profile` parameter in the command, the AWS CLI skips the environment variables and looks in the configuration file, which lets you override the credentials in the environment variables if you need to.)

The following example shows how you might set the environment variables for temporary security credentials and then call an AWS CLI command. Because no `profile` parameter is included in the AWS CLI command, the AWS CLI looks for credentials first in environment variables and therefore uses the temporary credentials.

Linux

```
$ export AWS_ACCESS_KEY_ID=AKIAI44QH8DHBEEXAMPLE
$ export AWS_SECRET_ACCESS_KEY=wJalrXutnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

```
$ export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of security token>
$ aws ec2 describe-instances --region us-west-1
```

Windows

```
C:\> SET AWS_ACCESS_KEY_ID=AKIAI44QH8DHBEEXAMPLE
C:\> SET AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
C:\> SET AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of token>
C:\> aws ec2 describe-instances --region us-west-1
```

Using Temporary Security Credentials with the Tools for Windows PowerShell

You can use temporary security credentials with the AWS Tools for Windows PowerShell. This can be useful for testing policies.

With the Tools for Windows PowerShell, you can call an AWS STS API action like `AssumeRole` or `GetFederationToken` and then capture the resulting output. The following example shows a PowerShell command that calls `AssumeRole` and stores the resulting role object in the variable `$role`. The `StoredCredentials` parameter is assumed to be a profile in the Tools for Windows PowerShell configuration file set up by `Set-AWSCredentials` or `Initialize-AWSDefaults` and is assumed to reference credentials for an IAM user who has permissions to assume the role.

```
PS C:\> $role = Use-STSRole -RoleArn arn:aws:iam::123456789012:role/MySampleRole -
RoleSessionName RoleSession1 -StoredCredentials IAM-user-name
```

When the command is finished, you can extract the access key ID, secret access key, and session token from the variable.

```
PS C:\> $role.AssumedRoleUser
Arn                                         AssumedRoleId
---                                         -----
arn:aws:sts::123456789012:assumed-role/clirole/RoleSession1
AROAJVIFQ5TJISRUTVTUE:RoleSession1

PS C:\> $role.Credentials.AccessKeyId
AKIAIOSFODNN7EXAMPLE

PS C:\> $role.Credentials.SecretAccessKey
wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

PS C:\> $role.Credentials.SessionToken
AQoDYXdzEPT///////////
wEXAMPLEtc764bNrC9SAPBSM22wD0k4x4HIZ8j4FZTwdQWLWsKWHGBuFqwAeMicRXmxfpSPfIeoIYRqT
f1fKD8YUuwthAx7mSEI/qkPpKPi/
kMcGdQrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDyOKPkyQDYwT7WZ0wq5VSXDvp75YU9
HFv1Rd8Tx6q6fE8YQcHNVXAkiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL64lIZbqBAz
+scqKmlzm8FDrypNC9Yjc8fPOLn9FX9KSYv
KTr4rvx3iS1lTJabIQwj2ICCR/oLxBA==

PS C:\> $role.Credentials.Expiration
Wednesday, July 08, 2015 3:22:32 PM
```

The following example shows how you might use the variable-stored credentials and then use them to call an AWS CLI command.

```
PS C:\> Get-EC2Instance -Region us-west-2 -Credential $role.Credentials
```

Using Temporary Security Credentials with APIs

If you're making direct HTTPS API requests to AWS, you can sign those requests with the temporary security credentials that you get from the AWS Security Token Service (AWS STS). To do this, you use the access key ID and secret access key that you receive from AWS STS the same way you would use long-term credentials to sign a request. You also add to your API request the session token that you receive from AWS STS. You add the session token to an HTTP header or to a query string parameter named `x-Amz-Security-Token`. You add the session token to the HTTP header or the query string parameter, but not both. For more information about signing HTTPS API requests, see [Signing AWS API Requests](#) in the *AWS General Reference*.

More Information

For more information about using AWS STS with other AWS services, see the following links:

- **Amazon S3.** See [Making Requests Using IAM User Temporary Credentials](#) or [Making Requests Using Federated User Temporary Credentials](#) in the *Amazon Simple Storage Service Developer Guide*.
- **Amazon SNS.** See [Using Temporary Security Credentials](#) in the *Amazon Simple Notification Service Developer Guide*.
- **Amazon SQS.** See [Using Temporary Security Credentials](#) in the *Amazon Simple Queue Service Developer Guide*.
- **Amazon SimpleDB.** See [Using Temporary Security Credentials](#) in the *Amazon SimpleDB Developer Guide*.

Controlling Permissions for Temporary Security Credentials

You can use AWS Security Token Service (AWS STS) to create and provide trusted users with temporary security credentials that can control access to your AWS resources. For more information about AWS STS, see [Temporary Security Credentials \(p. 231\)](#). After AWS STS issues temporary security credentials, they are valid through the expiration period and cannot be revoked. However, the permissions assigned to temporary security credentials are evaluated each time a request is made that uses the credentials, so you can achieve the effect of revoking the credentials by changing their access rights after they have been issued.

The following topics assume you have a working knowledge of AWS permissions and policies. For more information on these topics, see [Access Management \(p. 274\)](#).

Topics

- [Permissions for AssumeRole, AssumeRoleWithSAML, and AssumeRoleWithWebIdentity \(p. 246\)](#)
- [Permissions for GetFederationToken \(p. 249\)](#)
- [Permissions for GetSessionToken \(p. 252\)](#)
- [Disabling Permissions for Temporary Security Credentials \(p. 253\)](#)
- [Granting Permissions to Create Temporary Security Credentials \(p. 256\)](#)

Permissions for AssumeRole, AssumeRoleWithSAML, and AssumeRoleWithWebIdentity

The permission policy of the role that is being assumed determines the permissions for the temporary security credentials returned by `AssumeRole`, `AssumeRoleWithSAML`, and `AssumeRoleWithWebIdentity`. You define these permissions when you create or update the role.

Optionally, you can pass a separate policy as a parameter of the `AssumeRole`, `AssumeRoleWithSAML`, or `AssumeRoleWithWebIdentity` API call. You use the passed policy to scope down the permissions assigned to the temporary security credentials—that is, to allow only a subset of the permissions that are allowed by the permission policy of the role. You cannot use the passed policy to grant permissions that are in excess of those allowed by the permission policy of the role; it is a filter only. If you do not pass a policy as a parameter of the `AssumeRole`, `AssumeRoleWithSAML`, or `AssumeRoleWithWebIdentity` API call, then the permissions attached to the temporary security credentials are the same as the permissions defined in the role permissions policy of the role that is being assumed.

When the temporary security credentials returned by `AssumeRole`, `AssumeRoleWithSAML`, and `AssumeRoleWithWebIdentity` are used to make AWS requests, AWS determines whether to allow or deny access by evaluating all of the following policies:

- The combination of the role permission policy of the role that was assumed and the policy passed as a parameter to the API, as discussed previously.
- Any resource-based policies (such as an Amazon S3 bucket policy) attached to the resource that is being accessed by the temporary security credentials.

AWS uses all of these policies to arrive at an "allow" or "deny" authorization decision that follows the [IAM policy evaluation logic \(p. 458\)](#).

It is important to understand that the policies that are attached to the IAM user or the credentials that made the original call to `AssumeRole` are not evaluated by AWS when making the "allow" or "deny" authorization decision. The user temporarily gives up its original permissions in favor of the permissions assigned by the assumed role. In the case of the `AssumeRoleWithSAML` and `AssumeRoleWithWebIdentity` APIs, there are no policies to evaluate because the caller of the API is not an AWS identity.

Example: Assigning Permissions Using AssumeRole

You can use the `AssumeRole` API action with different kinds of policies. Here are a few examples.

Role Permission Policy

In this example, you call the `AssumeRole` API and do not include the optional `Policy` parameter. The permissions assigned to the temporary security credentials that are returned by the call to `AssumeRole`—that is, the permissions assigned to the *assumed role user*—are determined by the permission policy of the role being assumed. The following example is a role permission policy that grants the assumed role user permission to list all objects contained in an S3 bucket named `productionapp`, and to get, put, and delete objects within that bucket.

Example Role Permission Policy

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "s3>ListBucket",  
      "Resource": "arn:aws:s3:::productionapp"  
    },  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:GetObject",  
        "s3:PutObject",  
        "s3>DeleteObject"  
      ],  
      "Resource": "arn:aws:s3:::productionapp/*"  
    }  
  ]}
```

```
    ]  
}
```

Policy Passed as a Parameter

Imagine that you want to allow a user to assume the same role as in the previous example, but you want the assumed role user to have permission only to get and put objects—but not delete objects—in the productionapp S3 bucket. One way to accomplish this is to create a new role and specify the desired permissions in that role's permission policy. Another way to accomplish this is to call the `AssumeRole` API and include the optional `Policy` parameter as part of the API call. For example, imagine that the following policy is passed as a parameter of the API call. The assumed role user would have permissions to perform only these actions:

- List all objects in the productionapp bucket.
- Get and put objects in the productionapp bucket.

Note that because the `s3:DeleteObject` permission is not specified in the following policy, it is filtered out and the assumed role user is not granted the `s3:DeleteObject` permission. When you pass a policy as a parameter of the `AssumeRole` API call, the effective permissions of the assumed role user consist of only those permissions that are granted both in the role permission policy *and* the passed policy.

Example Policy Passed with `AssumeRole` API call

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "s3>ListBucket",  
      "Resource": "arn:aws:s3:::productionapp"  
    },  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:GetObject",  
        "s3:PutObject"  
      ],  
      "Resource": "arn:aws:s3:::productionapp/*"  
    }  
  ]  
}
```

Resource-based Policy

Some AWS resources support resource-based policies, and these policies provide another mechanism to define permissions that affect temporary security credentials. Only a few resources, like Amazon S3 buckets, Amazon SNS topics, and Amazon SQS queues support resource-based policies. The following example expands on the previous examples, using an S3 bucket named `productionapp`. The following policy is attached to the bucket.

When you attach the following policy to the `productionapp` bucket, *all* users are denied permission to delete objects from the bucket (note the `Principal` element in the policy). This includes all assumed role users, even though the role permission policy grants the `DeleteObject` permission. An explicit `Deny` statement always takes precedence over an explicit `Allow` statement.

Example Bucket Policy

```
{
```

```

    "Version": "2012-10-17",
    "Statement": [
        {
            "Principal": {"AWS": "*"},
            "Effect": "Deny",
            "Action": "s3:DeleteObject",
            "Resource": "arn:aws:s3:::productionapp/*"
        }
    ]
}

```

For more information about how multiple policies are combined and evaluated by AWS, see [IAM JSON Policy Evaluation Logic \(p. 458\)](#).

Permissions for GetFederationToken

The permissions for the temporary security credentials returned by `GetFederationToken` are determined by a combination of the following:

- The policy or policies that are attached to the IAM user whose credentials are used to call `GetFederationToken`.
- The policy that is passed as a parameter in the call.

The passed policy is attached to the temporary security credentials that result from the `GetFederationToken` API call—that is, to the *federated user*. When the federated user makes an AWS request, AWS evaluates the policy attached to the federated user in combination with the policy or policies attached to the IAM user whose credentials were used to call `GetFederationToken`.

Important

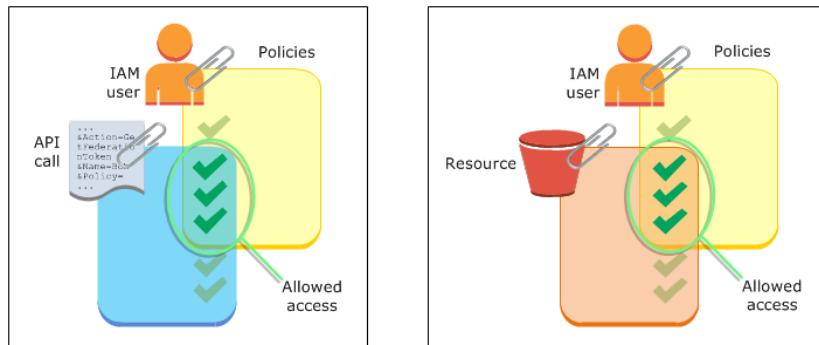
AWS allows the federated user's request only when both the attached policy *and* the IAM user policy explicitly allow the federated user to perform the requested action.

The permissions assigned to the federated user are defined in one of two places:

- The policy passed as a parameter of the `GetFederationToken` API call. (This is most common.)
- A resource-based policy that explicitly names the federated user in the `Principal` element of the policy. (This is less common.)

This means that in most cases if you do not pass a policy with the `GetFederationToken` API call, the resulting temporary security credentials have no permissions. The only exception is when the credentials are used to access a resource that has a resource-based policy that specifically references the federated user in the `Principal` element of the policy.

The following figures show a visual representation of how the policies interact to determine permissions for the temporary security credentials returned by a call to `GetFederationToken`.



Example: Assigning Permissions Using `GetFederationToken`

You can use the `GetFederationToken` API action with different kinds of policies. Here are a few examples.

Policy Attached to the IAM User

In this example, you have a browser-based client application that relies on two back-end web services. One back-end service is your own authentication server that uses your own identity system to authenticate the client application. The other back-end service is an AWS service that provides some of the client application's functionality. The client application is authenticated by your server, and your server creates or retrieves the appropriate permission policy. Your server then calls the `GetFederationToken` API to obtain temporary security credentials, and returns those credentials to the client application. The client application can then make requests directly to the AWS service with the temporary security credentials. This architecture allows the client application to make AWS requests without embedding long-term AWS credentials.

Your authentication server calls the `GetFederationToken` API with the long-term security credentials of an IAM user named `token-app`, but the long-term IAM user credentials remain on your server and are never distributed to the client. The following example policy is attached to the `token-app` IAM user and defines the broadest set of permissions that your federated users (clients) will need. Note that the `sts:GetFederationToken` permission is required for your authentication service to obtain temporary security credentials for the federated users.

Note

AWS provides a sample Java application to serve this purpose, which you can download here:
[Token Vending Machine for Identity Registration - Sample Java Web Application](#).

Example Policy Attached to IAM User `token-app` that Calls `GetFederationToken`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:GetFederationToken",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "sns:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "lambda:*",
            "Resource": "*"
        }
    ]
}
```

Although the preceding policy grants several permissions, by itself it is not enough to grant any permissions to the federated user. If the IAM user that has the permissions defined in the preceding policy calls `GetFederationToken` and does not pass a policy as a parameter of the API call, the resulting federated user has no effective permissions.

Policy Passed as Parameter

The most common way to ensure that the federated user is assigned appropriate permission is to pass a policy as a parameter of the `GetFederationToken` API call. Expanding on our previous example, imagine that `GetFederationToken` is called with the credentials of the IAM user `token-app` and imagine that the following policy is passed as a parameter of the API call. The federated user would have permission to perform only these actions:

- List the contents of the Amazon S3 bucket named `productionapp`.
- Perform the Amazon S3 `GetObject`, `PutObject`, and `DeleteObject` actions on items in the `productionapp` bucket.

The federated user is assigned these permissions because the permissions have been granted to both the IAM user who called `GetFederationToken` (via the policy attached to the IAM user) *and* to the federated user (via the passed policy). The federated user could not perform actions in Amazon SNS, Amazon SQS, Amazon DynamoDB, or in any S3 bucket except `productionapp`, even though those permissions are granted to the IAM user associated with the `GetFederationToken` call. This is true because the effective permissions for the federated user consist of only those permissions that are granted in both the IAM user policy *and* the passed policy.

Example Policy Passed as Parameter of `GetFederationToken` API call

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": ["s3>ListBucket"],  
            "Resource": ["arn:aws:s3:::productionapp"]  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:GetObject",  
                "s3:PutObject",  
                "s3:DeleteObject"  
            ],  
            "Resource": ["arn:aws:s3:::productionapp/*"]  
        }  
    ]  
}
```

Resource-based Policies

Some AWS resources support resource-based policies, and these policies provide another mechanism to grant permissions directly to a federated user. Only some AWS services support resource-based policies. For example, Amazon S3 has buckets, Amazon SNS has topics, and Amazon SQS has queues that you can attach policies to. For a list of all services that support resource-based policies, see [AWS Services That Work with IAM \(p. 417\)](#) and review the "Resource Based" column of the tables. If you use one of these services and resource-based policies makes sense for your scenario, you assign permissions directly to a federated user by specifying the Amazon Resource Name of the federated user in the `Principal` element of the resource-based policy. The following example illustrates this. The following example expands on the previous examples, using an S3 bucket named `productionapp`.

The following policy is attached to the bucket. The policy allows a federated user named Carol to access the bucket. When the following resource-based policy is in place, and the example policy described earlier is attached to IAM user token-app, the federated user named Carol has permission to perform the s3:GetObject, s3:PutObject, and s3:DeleteObject actions on the bucket named productionapp. This is true even when no policy is passed as a parameter of the GetFederationToken API call. That's because in this case the federated user named Carol has been explicitly granted permissions by the following resource-based policy.

Remember, a federated user is granted permissions only when those permissions are explicitly granted to both the IAM user **and** the federated user. Permissions can be granted to the federated user by the policy passed as a parameter of the GetFederationToken API call, or by a resource-based policy that explicitly names the federated user in the Principal element of the policy, as in the following example.

Example Bucket Policy that Allows Access to Federated User

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Principal": {"AWS": "arn:aws:sts::ACCOUNT-ID-WITHOUT-HYPHENS:federated-user/Carol"},  
            "Effect": "Allow",  
            "Action": [  
                "s3:GetObject",  
                "s3:PutObject",  
                "s3:DeleteObject"  
            ],  
            "Resource": ["arn:aws:s3:::productionapp/*"]  
        }  
    ]  
}
```

Permissions for GetSessionToken

If GetSessionToken is called with the credentials of an IAM user, the temporary security credentials have the same permissions as the IAM user. Similarly, if GetSessionToken is called with AWS account root user credentials, the temporary security credentials have root user permissions.

Note

We recommend that you do not call GetSessionToken with root user credentials. Instead, follow our [best practices \(p. 43\)](#) and create one or more IAM users with the permissions they need. Then use these IAM users for everyday interaction with AWS.

The primary reason for calling GetSessionToken is that a user needs to perform actions that are allowed only when the user is authenticated with multi-factor authentication (MFA). It is possible to write a policy that allows certain actions only when those actions are requested by a user who has been authenticated with MFA. In order to successfully pass the MFA authorization check, a user must first call GetSessionToken and include the optional SerialNumber and TokenCode parameters in the GetSessionToken API call. If the user passes valid values for the SerialNumber and TokenCode parameters—that is, if the user is successfully authenticated with an MFA device—the credentials returned by the GetSessionToken API call will include the MFA context. Subsequent calls to AWS with the credentials returned by GetSessionToken will allow the user to successfully call AWS APIs that require MFA authentication.

The temporary credentials that you get when you call GetSessionToken have the following capabilities and limitations:

- You can use the credentials to access the AWS Management Console by passing the credentials to the federation single sign-on endpoint at <https://signin.aws.amazon.com/federation>. For more information, see [Creating a URL that Enables Federated Users to Access the AWS Management Console \(Custom Federation Broker\) \(p. 169\)](#).

- You **cannot** use the credentials to call IAM or AWS STS APIs. You **can** use them to call APIs for other AWS services.

Compare this API and its limitations and capability with the other APIs that create temporary security credentials at [Comparing the AWS STS APIs \(p. 241\)](#)

For more information about MFA-protected API access using `GetSessionToken`, see [Configuring MFA-Protected API Access \(p. 107\)](#).

Disabling Permissions for Temporary Security Credentials

Temporary security credentials are valid until they expire, and they cannot be revoked. However, because permissions are evaluated each time an AWS request is made using the credentials, you can achieve the effect of revoking the credentials by changing the permissions for the credentials even after they have been issued. If you remove all permissions from the temporary security credentials, subsequent AWS requests that use those credentials will fail. The mechanisms for changing or removing the permissions assigned to temporary security credentials are explained in the following sections.

Note

When you update existing policy permissions, or when you apply a new policy to a user or a resource, it may take a few minutes for policy updates to take effect.

Topics

- [Denying Access to the Creator of the Temporary Security Credentials \(p. 253\)](#)
- [Denying Access to Temporary Security Credentials by Name \(p. 254\)](#)
- [Denying Access to Temporary Security Credentials Issued Before a Specific Time \(p. 255\)](#)

Denying Access to the Creator of the Temporary Security Credentials

To change or remove the permissions assigned to temporary security credentials, you can change or remove the permissions that are associated with the creator of the credentials. The creator of the credentials is determined by the AWS STS API that was used to obtain the credentials. The mechanisms for changing or removing the permissions associated with this creator are explained in the following sections.

[Denying Access to Credentials Created by AssumeRole, AssumeRoleWithSAML, or AssumeRoleWithWebIdentity](#)

To change or remove the permissions assigned to the temporary security credentials obtained by calling the `AssumeRole`, `AssumeRoleWithSAML`, or `AssumeRoleWithWebIdentity` APIs, you edit or delete the role permission policy that defines the permissions for the assumed role. The temporary security credentials obtained by assuming a role can never have more permissions than those defined in the access policy of the assumed role, and the permissions assigned to temporary security credentials are evaluated each time they are used to make an AWS request. When you edit or delete the permission policy of a role, the changes affect the permissions of *all* temporary security credentials associated with that role, including credentials that were issued before you changed the role's access policy. You can immediately revoke all permissions to a session by following the steps at [Revoking IAM Role Temporary Security Credentials \(p. 220\)](#).

For more information about editing a role permission policy, see [Modifying a Role \(p. 222\)](#).

[Denying Access to Credentials Created by GetFederationToken or GetSessionToken](#)

To change or remove the permissions assigned to the temporary security credentials obtained by calling the `GetFederationToken` or `GetSessionToken` APIs, you edit or delete the policies that are attached to the IAM user whose credentials were used to call `GetFederationToken` or `GetSessionToken`.

The temporary security credentials that were obtained by calling `GetFederationToken` or `GetSessionToken` can never have more permissions than the IAM user whose credentials were used to obtain them. In addition the permissions assigned to temporary security credentials are evaluated each time they are used to make an AWS request. It is important to note that when you edit or delete the permissions of an IAM user, the changes affect the IAM user as well as all temporary security credentials created by that user.

Important

You cannot change the permissions for an AWS account root user. Likewise, you cannot change the permissions for the temporary security credentials that were created by calling `GetFederationToken` or `GetSessionToken` while signed in as the root user. For this reason, we recommend that you do not call `GetFederationToken` or `GetSessionToken` as a root user.

For information about how to change or remove the policies associated with the IAM user whose credentials were used to call `GetFederationToken` or `GetSessionToken`, see [Managing IAM Policies \(p. 316\)](#).

Denying Access to Temporary Security Credentials by Name

You can deny access to temporary security credentials without affecting the permissions of the IAM user or role that created the credentials. You do this by specifying the Amazon Resource Name (ARN) of the temporary security credentials in the `Principal` element of a resource-based policy. (Only some AWS services support resource-based policies.)

Denying Access to Federated Users

For example, imagine you have an IAM user named `token-app` whose credentials are used to call `GetFederationToken`. The `GetFederationToken` API call resulted in temporary security credentials associated with a federated user named Bob (the federated user's name is taken from the `Name` parameter of the API call). To deny federated user Bob's access to an S3 bucket called `EXAMPLE-BUCKET`, you attach the following example bucket policy to `EXAMPLE-BUCKET`. It is important to note that this affects the federated user's Amazon S3 permissions only—any other permissions granted to the federated user remain intact.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {"Principal": {"AWS": "arn:aws:sts::ACCOUNT-ID-WITHOUT-HYPHENS:federated-user/Bob"},  
         "Effect": "Deny",  
         "Action": "s3:*",  
         "Resource": "arn:aws:s3:::EXAMPLE-BUCKET"}  
    ]  
}
```

You can specify the ARN of the IAM user whose credentials were used to call `GetFederationToken` in the `Principal` element of the bucket policy, instead of specifying the federated user. In that case, the `Principal` element of the preceding policy would look like this:

```
"Principal": {"AWS": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:user/token-app"}
```

It is important to note that specifying the ARN of IAM user `token-app` in the policy will result in denying access to *all* federated users created by `token-app`, not only the federated user named Bob.

Denying Access to Assumed Role Users

It is also possible to specify the ARN of the temporary security credentials that were created by assuming a role. The difference is the syntax used in the `Principal` element of the resource-based policy. For

example, a user assumes a role called Accounting-Role and specifies a RoleSessionName of Mary (RoleSessionName is a parameter of the AssumeRole API call). To deny access to the temporary security credentials that resulted from this API call, the Principal element of the resource-based policy would look like this:

```
"Principal": {"AWS": "arn:aws:sts::ACCOUNT-ID-WITHOUT-HYPHENS:assumed-role/Accounting-Role/Mary"}
```

You can also specify the ARN of the IAM role in the Principal element of a resourced-based policy, as in the following example. In this case, the policy will result in denying access to *all* temporary security credentials associated with the role named Accounting-Role.

```
"Principal": {"AWS": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:role/Accounting-Role"}
```

Denying Access to Temporary Security Credentials Issued Before a Specific Time

It is possible to deny access only to temporary security credentials that were created before a specific time and date. You do this by specifying a value for the aws:TokenIssueTime key in the Condition element of a policy. The following policy shows an example. You attach a policy similar to the following example to the IAM user that created the temporary security credentials. The policy denies all permissions but only when the value of aws:TokenIssueTime is earlier than the specified date and time. The value of aws:TokenIssueTime corresponds to the exact time at which the temporary security credentials were created. The aws:TokenIssueTime value is only present in the context of AWS requests that are signed with temporary security credentials, so the Deny statement in the policy will not affect requests that are signed with the long-term credentials of the IAM user.

The following policy can also be attached to a role. In that case, the policy affects only the temporary security credentials that were created by the role before the specified date and time. If the credentials were created by the role after the specified date and time, the Condition element in the policy is evaluated to false, so the Deny statement has no effect.

Example Policy that Denies All Permissions to Temporary Credentials by Issue Time

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "*",
            "Resource": "*",
            "Condition": {"DateLessThan": {"aws:TokenIssueTime": "2014-05-07T23:47:00Z"}}
        }
    ]
}
```

Valid users whose sessions are revoked in this way must acquire temporary credentials for a new session to continue working. Note that the AWS CLI caches credentials until they expire. To force the CLI to delete and refresh cached credentials that are no longer valid, run one of the following commands:

Linux, MacOS, or Unix

```
$ rm -r ~/.aws/cli/cache
```

Windows

```
C:> del /s /q %UserProfile%\.aws\cli\cache
```

Granting Permissions to Create Temporary Security Credentials

By default, IAM users do not have permission to create temporary security credentials for federated users and roles. However, by default IAM users can call [GetSessionToken](#) with their own permissions and thereby create temporary credentials for others..

Note

Although you can grant permissions directly to a user, we strongly recommend that you grant permissions to a group. This makes management of the permissions much easier. When someone no longer needs to perform the tasks associated with the permissions, you simply remove them from the group. If someone else needs to perform that task, add them to the group to grant the permissions.

To grant an IAM group permission to create temporary security credentials for federated users or roles, you attach a policy that grants one or both of the following privileges:

- For federated users to access an IAM role, grant access to AWS STS `AssumeRole`.
- For federated users that don't need a role, grant access to AWS STS `GetFederationToken`.

For more information about the differences between the `AssumeRole` and `GetFederationToken` APIs, see [Requesting Temporary Security Credentials \(p. 233\)](#).

Example : A policy that grants permission to assume a role

The following example policy grants permission to call `AssumeRole` for the `UpdateAPP` role in AWS account 123123123123. When `AssumeRole` is used, the user (or application) that creates the security credentials on behalf of a federated user cannot delegate any permissions not already specified in the role permission policy.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "sts:AssumeRole",  
            "Resource": "arn:aws:iam::123123123123:role/UpdateAPP"  
        }  
    ]  
}
```

Example : A policy that grants permission to create temporary security credentials for a federated user

The following example policy grants permission to access `GetFederationToken`.

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

Important

When you give an IAM user permission to create temporary security credentials for federated users with `GetFederationToken`, be aware that this permits the IAM user to delegate his or her own permissions. For more information about delegating permissions across IAM users and

AWS accounts, see [Examples of Policies for Delegating Access \(p. 201\)](#). For more information about controlling permissions in temporary security credentials, see [Controlling Permissions for Temporary Security Credentials \(p. 246\)](#).

Example : A policy that grants a user limited permission to create temporary security credentials for federated users

When you let an IAM user call `GetFederationToken` to create temporary security credentials for federated users, it is a best practice to restrict as much as practical the permissions that the IAM user is allowed to delegate. For example, the following policy shows how to let an IAM user create temporary security credentials only for federated users whose names start with *Manager*.

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": "sts:GetFederationToken",  
        "Resource": ["arn:aws:sts::123456789012:federated-user/Manager*"]  
    }]  
}
```

Activating and Deactivating AWS STS in an AWS Region

By default, the AWS Security Token Service (AWS STS) is available as a global service, and all AWS STS requests go to a single endpoint at `https://sts.amazonaws.com`. You can optionally send your AWS STS requests to endpoints in any of the AWS regions shown in the table that follows. All regions are enabled by default, but you can disable any regions that you know you don't need.

You might choose to send your AWS STS requests to a regional endpoint for the following reasons:

- **Reduce latency** – By making your AWS STS calls to an endpoint that is geographically closer to your services and applications, you can access AWS STS services with lower latency and better response times.
- **Build in Redundancy** – By adding code to your application that switches your AWS STS API calls to a different region, you ensure that if the first region stops responding, your application continues to operate. This redundancy is not automatic; you must build the functionality into your code.

When you activate a region for an account, you enable the STS endpoints in that region to issue temporary credentials for users and roles in that account when a request is made to an endpoint in the region. The credentials are still recognized and usable globally. It is not the account of the caller, but the account from which temporary credentials are requested that must activate the region.

For example, imagine a user in account A wants to send an `sts:AssumeRole` API request to the STS regional endpoint `https://sts.us-west-2.amazonaws.com`. The request is for temporary credentials for the role named `Developer` in account B. Because the request is to create credentials for an entity in account B, account B must activate the `us-west-2` region. Users from account A (or any other account) can call the `us-west-2` endpoint to request credentials for account B whether or not the region is activated in their accounts.

To activate or deactivate AWS STS in a region

Note

All regions are activated by default. You need to activate a region only if it was previously deactivated.

1. Sign in as an IAM user with permissions to perform IAM administration tasks ("iam:*") for the account for which you want to activate AWS STS in a new region.
2. Open the IAM console and in the navigation pane choose [Account Settings](#).
3. Expand the **Security Token Service Regions** list, find the region that you want to use, and then choose **Activate** or **Deactivate**.

Writing Code to Use AWS STS Regions

After you activate a region for your AWS account, you can direct AWS STS API calls to that region. The following Java code snippet demonstrates how to configure an `AWSecurityTokenServiceClient` object to make requests from the EU (Ireland) (eu-west-1) region with the `setEndpoint` method.

```
AWSecurityTokenServiceClient stsClient = new AWSecurityTokenServiceClient();
stsClient.setEndpoint("sts.eu-west-1.amazonaws.com");
```

Important

Do not use the `setRegion` method to set a regional endpoint for AWS STS because, for backward compatibility, that method continues to resolve to the original single global endpoint of `sts.amazonaws.com`.

In the example, the first line instantiates an `AWSecurityTokenServiceClient` object called `stsClient`. The second line configures the `stsClient` object by calling its `setEndpoint` method and passing the URL of the endpoint as the only parameter. All API calls that use this `stsClient` object are now directed to the specified endpoint.

For all other language and programming environment combinations, refer to the [documentation for the relevant SDK](#).

Nothing else about using AWS STS in a region changes. As always, the credentials retrieved from the AWS STS endpoint in a region are not limited to that region; you can use them globally.

The following table lists the regions and their endpoints. It indicates which ones are activated by default and which ones you can activate or deactivate.

Region Name	Endpoint	Can be activated/ deactivated
--Global--	sts.amazonaws.com	No
US East (Ohio)	sts.us-east-2.amazonaws.com	Yes
US East (N. Virginia)	sts.us-east-1.amazonaws.com	No
US West (N. California)	sts.us-west-1.amazonaws.com	Yes
US West (Oregon)	sts.us-west-2.amazonaws.com	Yes
Canada (Central)	sts.ca-central-1.amazonaws.com	Yes
Asia Pacific (Tokyo)	sts.ap-northeast-1.amazonaws.com	Yes
Asia Pacific (Seoul)	sts.ap-northeast-2.amazonaws.com	Yes
Asia Pacific (Mumbai)	sts.ap-south-1.amazonaws.com	Yes
Asia Pacific (Singapore)	sts.ap-southeast-1.amazonaws.com	Yes

Region Name	Endpoint	Can be activated/deactivated
Asia Pacific (Sydney)	sts.ap-southeast-2.amazonaws.com	Yes
EU (Frankfurt)	sts.eu-central-1.amazonaws.com	Yes
EU (Ireland)	sts.eu-west-1.amazonaws.com	Yes
EU (London)	sts.eu-west-2.amazonaws.com	Yes
South America (São Paulo)	sts.sa-east-1.amazonaws.com	Yes

Note

Calls to regional endpoints, such as `us-east-2.amazonaws.com`, are logged in AWS CloudTrail the same as any call to a regional service. Calls to the global endpoint, `sts.amazonaws.com`, are logged as calls to a global service. For more information, see [Logging IAM Events with AWS CloudTrail \(p. 262\)](#).

Sample Applications That Use Temporary Credentials

You can use AWS Security Token Service (AWS STS) to create and provide trusted users with temporary security credentials that can control access to your AWS resources. For more information about AWS STS, see [Temporary Security Credentials \(p. 231\)](#). To see how you can use AWS STS to manage temporary security credentials, you can download the following sample applications that implement complete example scenarios:

- [Identity Federation Sample Application for an Active Directory Use Case](#). Demonstrates how to use permissions that are tied to a user defined in Active Directory (.NET/C#) to issue temporary security credentials for accessing Amazon S3 files and buckets.
- [AWS Management Console Federation Proxy Sample Use Case](#). Demonstrates how to create a custom federation proxy that enables single sign-on (SSO) so that existing Active Directory users can sign into the AWS Management Console (.NET/C#).
- [Integrate Shibboleth with AWS Identity and Access Management](#). Shows how to use [Shibboleth](#) and [SAML \(p. 149\)](#) to provide users with single sign-on (SSO) access to the AWS Management Console.

Samples for Web Identity Federation

The following sample applications illustrate how to use web identity federation with providers like Login with Amazon, Amazon Cognito, Facebook, or Google. You can trade authentication from these providers for temporary AWS security credentials to access AWS services.

Note

As an alternative, we recommend that you use Amazon Cognito with the AWS SDKs for mobile development. Amazon Cognito is the simplest way to manage identity for mobile apps, and it provides additional features like synchronization and cross-device identity. For more information about Amazon Cognito, see [Amazon Cognito Identity](#) in the [AWS Mobile SDK for Android Developer Guide](#) and [Authenticate Users with Amazon Cognito Identity](#) in the [AWS Mobile SDK for iOS Developer Guide](#).

- [Web Identity Federation Playground](#). This website provides an interactive demonstration of [web identity federation \(p. 143\)](#) and the `AssumeRoleWithWebIdentity` API.
- [Build and Deploy a Federated Web Identity Application with AWS Elastic Beanstalk and Login with Amazon](#). This blog post describes how to use `AssumeRoleWithWebIdentity` to obtain temporary

security credentials through web identity federation and Login with Amazon. It also explains how to use those credentials in a Python web application that runs on Elastic Beanstalk to make calls to AWS.

- [Authenticating Users of AWS Mobile Applications with a Token Vending Machine at AWS Articles & Tutorials](#). This sample demonstrates a server-based proxy application that serves temporary credentials to remote clients (such as mobile apps) so that the clients can sign web requests to AWS. This sample can be used with the sample client that is part of the AWS Mobile SDK for Android and the AWS Mobile SDK for iOS. For more information, see [Credential Management for Mobile Applications](#), which provides details on how to secure AWS resources when you use the token vending machine (TVM) with mobile applications.

Additional Resources for Temporary Security Credentials

The following scenarios and applications can guide you in using temporary security credentials:

- [About Web Identity Federation \(p. 143\)](#). This section discusses how to configure IAM roles when you use web identity federation and the `AssumeRoleWithWebIdentity` API.
- [Configuring MFA-Protected API Access \(p. 107\)](#). This topic explains how to use roles to require multi-factor authentication (MFA) to protect sensitive API actions in your account.
- [Token Vending Machine for Identity Registration](#). This sample Java web application uses the `GetFederationToken` API to serve temporary security credentials to remote clients.

For more information on policies and permissions in AWS see the following topics:

- [Access Management \(p. 274\)](#)
- [IAM JSON Policy Evaluation Logic \(p. 458\)](#).
- [Managing Access Permissions to Your Amazon S3 Resources in Amazon Simple Storage Service Developer Guide](#).

The AWS Account Root User

When you first create an Amazon Web Services (AWS) account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account.

Important

We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the [best practice of using the root user only to create your first IAM user \(p. 44\)](#). Then securely lock away the root user credentials and use them to perform only a few account and service management tasks. To view the tasks that require you to sign in as the root user, see [AWS Tasks That Require Root User](#). For a tutorial on how to set up an administrator for daily use, see [Creating Your First IAM Admin User and Group \(p. 17\)](#).

To manage your root user, follow the steps in the following procedures.

Topics

- [Enable MFA on the AWS Account Root User \(p. 261\)](#)
- [Creating Access Keys for the Root User \(p. 261\)](#)
- [Deleting Access Keys from the Root User \(p. 262\)](#)

- [Changing the Root User's Password \(p. 262\)](#)

Enable MFA on the AWS Account Root User

If you continue to use the root user credentials, it is a security best practice to enable multi-factor authentication (MFA) for your account. Because your root user can perform sensitive operations in your account, adding an additional layer of authentication helps you to better secure your account. Multiple types of MFA are available. For more information about enabling MFA, see the following:

- [Enable a Virtual MFA Device for Your AWS Account Root User \(Console\) \(p. 94\)](#)
- [Enable a Hardware MFA Device for the AWS Account Root User \(Console\) \(p. 97\)](#)

Creating Access Keys for the Root User

You can use the AWS Management Console or AWS programming tools to create access keys for the root user.

To create an access key for the AWS account root user (console)

1. Use your AWS account email address and password to sign in to the [AWS Management Console](#) as the AWS account root user.

Note

If you previously signed in to the console with [IAM user](#) credentials, your browser might remember this preference and open your account-specific sign-in page. You cannot use the IAM user sign-in page to sign in with your AWS account root user credentials. If you see the IAM user sign-in page, choose **Sign-in using root user credentials** near the bottom of the page to return to the main sign-in page. From there, you can type your AWS account email address and password.

2. On the **IAM Dashboard** page, choose your account name in the navigation bar, and then choose **My Security Credentials**.
3. If you see a warning about accessing the security credentials for your AWS account, choose **Continue to Security Credentials**.
4. Expand the **Access keys (access key ID and secret access key)** section.
5. Choose **Create New Access Key**. A warning explains that you have only this one opportunity to view or download the secret access key. It cannot be retrieved later.
 - If you choose **Show Access Key**, you can copy the access key ID and secret key from your browser window and paste it somewhere else.
 - If you choose **Download Key File**, you receive a file named `rootkey.csv` that contains the access key ID and the secret key. Save the file somewhere safe.
6. When you no longer use the access key [we recommend that you delete it \(p. 47\)](#), or at least mark it inactive by choosing **Make Inactive** so that it cannot be misused if leaked.

To create an access key for the root user (programmatically)

Use one of the following commands:

- AWS CLI: `aws iam create-access-key`
- Tools for Windows PowerShell: `New-IAMAccessKey`
- AWS API: `CreateAccessKey`

Deleting Access Keys from the Root User

You can use the AWS Management Console or various programming tools to delete access keys for the root user.

To delete an access key from the AWS account root user (console)

1. Use your AWS account email address and password to sign in to the [AWS Management Console](#) as the root user.

Note

If you previously signed in to the console with [IAM user \(p. 61\)](#) credentials, your browser might remember this preference and open your account-specific sign-in page. You cannot use the IAM user sign-in page to sign in with your AWS account root user credentials. If you see the IAM user sign-in page, choose **Sign-in using root account credentials** near the bottom of the page to return to the main sign-in page. From there, you can type your AWS account email address and password.

2. On the **IAM Dashboard** page, choose your account name in the navigation bar, and then choose **My Security Credentials**.
3. If you see a warning about accessing the security credentials for your AWS account, choose **Continue to Security Credentials**.
4. Expand the **Access keys (access key ID and secret access key)** section.
5. Find the access key that you want to delete, and then, under the **Actions** column, choose **Delete**.

Note

You can mark an access key as inactive instead of deleting it. This enables you to resume use of it in the future without having to change either the key ID or secret key. While it is inactive, any attempts to use it in requests to the AWS API fail with the status of access denied.

To delete an access key for the root user (programmatically)

Use one of the following commands:

- AWS CLI: `aws iam delete-access-key`
- Tools for Windows PowerShell: `Remove-IAMAccessKey`
- AWS API: `DeleteAccessKey`

Changing the Root User's Password

For information about changing the root user's password, see [Changing the AWS Account Root User Password \(p. 75\)](#). To change the root user, you must log in using the root user credentials. To view the tasks that require you to sign in as the root user, see [AWS Tasks that Require Root User](#)

Logging IAM Events with AWS CloudTrail

AWS Identity and Access Management (IAM) is integrated with AWS CloudTrail, a service that logs AWS events made by or on behalf of your AWS account. CloudTrail logs authenticated AWS API calls and also AWS sign-in events, and collects this event information in files that are delivered to Amazon S3 buckets. Using information collected by CloudTrail, you can determine what requests were successfully made to AWS services, who made the request, when it was made, and so on.

To learn more about CloudTrail, including how to configure and enable it, see the [AWS CloudTrail User Guide](#).

Topics

- [Types of IAM Information Logged in CloudTrail \(p. 263\)](#)
- [Examples of Logged Events in CloudTrail Files \(p. 265\)](#)
- [Preventing Duplicate Log Entries in CloudTrail \(p. 271\)](#)

Types of IAM Information Logged in CloudTrail

IAM information is available to CloudTrail in these ways:

- **API requests to IAM and AWS Security Token Service (AWS STS)** – CloudTrail logs all authenticated API requests (made with credentials) to IAM and AWS STS APIs, with the exception of `DecodeAuthorizationMessage`. CloudTrail also logs nonauthenticated requests to the AWS STS actions, `AssumeRoleWithSAML` and `AssumeRoleWithWebIdentity` and logs information provided by the identity provider. You can use this information to map calls made by a federated user with an assumed role back to the originating external federated caller. In the case of `AssumeRole`, you can map calls back to the originating AWS service or to the account of the originating user. The `userIdentity` section of the JSON data in the CloudTrail log entry contains the information that you need to map the `AssumeRole*` request with a specific federated user. For more information, see [CloudTrail userIdentity Element](#) in the [AWS CloudTrail User Guide](#).

For example, calls to the IAM `CreateUser`, `DeleteRole`, `ListGroups`, and other API operations are all logged by CloudTrail.

Examples for this type of log entry are presented later in this topic.

Important

If you activate AWS STS endpoints in regions other than the default global endpoint, then you must also turn on CloudTrail logging in those regions to record any AWS STS API calls made in those regions. For more information, see [Turning On CloudTrail in Additional Regions](#) in the [AWS CloudTrail User Guide](#).

- **API requests to other AWS services** – Authenticated requests to other AWS service APIs are logged by CloudTrail, and these log entries contain information about who generated the request.

For example, if a request is made to list Amazon EC2 instances or create an AWS CodeDeploy deployment group, the user identity of the person or service that made the request is contained in the log entry for that request. The user identity information helps you determine whether the request was made with AWS account root user credentials or IAM user credentials, with temporary security credentials for a role or federated user, or by another AWS service.

For more details about the user identity information in CloudTrail log entries, see [userIdentity Element](#) in the [AWS CloudTrail User Guide](#).

- **AWS sign-in events** – Sign-in events to the AWS Management Console, the AWS Discussion Forums, and the AWS Marketplace are logged by CloudTrail.

For example, IAM and federated user sign-in events—successful sign-ins and failed sign-in attempts—are logged by CloudTrail. Additionally, successful sign-in events by the root user are logged by CloudTrail. Note that unsuccessful sign-in events by the root user are *not* logged by CloudTrail.

If you enable CloudTrail to log sign-in events to your logs, you need to be aware of how CloudTrail chooses where to log the events.

- If your users sign in directly to a console, they are redirected to either a global or a regional sign-in endpoint, based on whether the selected service console supports regions. For example, the main console home page supports regions, so if you sign in to the following URL:

```
https://alias.signin.aws.amazon.com/console
```

you are redirected to a regional sign-in endpoint such as `https://us-east-2.signin.aws.amazon.com`, resulting in a regional CloudTrail log entry in the user's region's log:

On the other hand, the Amazon S3 console does not support regions, so if you sign in to the following URL

```
https://alias.signin.aws.amazon.com/console/s3
```

AWS redirects you to the global sign-in endpoint at `https://signin.aws.amazon.com`, resulting in a global CloudTrail log entry.

- You can manually request a certain regional sign-in endpoint by signing in to the region-enabled main console home page using a URL syntax like the following:

```
https://alias.signin.aws.amazon.com/console?region=ap-southeast-1
```

AWS redirects you to the `ap-southeast-1` regional sign-in endpoint and results in a regional CloudTrail log event.

Important

As a security best practice, AWS does not log the entered user name text when the sign-in failure is caused by *an incorrect user name*. The user name text is masked by the value `HIDDEN_DUE_TO_SECURITY_REASONS`. For an example of this, see [Sign-in Failure Event Caused by Incorrect User Name \(p. 270\)](#), later in this topic. The reason the user name is obscured is because such failures might be caused by user errors like the following, which, if logged, could expose potentially sensitive information:

- You accidentally type your password in the user name field.
- You click the link for one AWS account's sign-in page, but then type the account number for a different one.
- You forget which account you are signing in to and accidentally type the account name of your personal email account, your bank sign-in identifier, or some other private ID.

Whether the sign-in event is considered to a regional event or a global one depends on the console the user is signing into, and how the user constructs the sign-in URL.

- Is the service console regionalized? If so, then the sign-in request is automatically redirected to a regional sign-in endpoint and the event is logged in that region's CloudTrail log. For example, if you sign in to `https://alias.signin.aws.amazon.com/console` the console home page which is regionalized, you are automatically redirected you to a sign-in endpoint in your region (for example, `https://us-east-2.signin.aws.amazon.com`), and the event is logged in that region's log.

However, some services are not regionalized yet. For example, the Amazon S3 service is *not* currently regionalized, so if you sign in to `https://alias.signin.aws.amazon.com/console/s3`, you are redirected to the global sign-in endpoint at `https://signin.aws.amazon.com`, resulting in an event in your global log.

- You can also manually request a certain regional sign-in endpoint by using a URL syntax like `https://alias.signin.aws.amazon.com/console?region=ap-southeast-1`, which redirects to the `ap-southeast-1` regional sign-in endpoint and results in an event in the regional log.
- **How temporary credential requests are logged** – When a principal requests temporary credentials, the principal type determines how CloudTrail logs the event. The following table shows how CloudTrail logs different information for each of the API calls that generate temporary credentials.

Principal Type	IAM/STS API	User Identity in CloudTrail Log for Calling Account	User Identity in CloudTrail Log for Role Owning Account	User Identity in CloudTrail Log for Role Owner for Subsequent API calls
AWS account root user credentials	GetSessionToken	Root identity	Role owner account is same as calling account	Root identity
IAM user	GetSessionToken	IAM user identity	Role owner account is same as calling account	IAM user identity
IAM user	GetFederationToken	IAM user identity	Role owner account is same as calling account	IAM user identity
IAM user	AssumeRole	IAM user identity	Account number and Principal ID (if a user), or AWS service principal	Role identity only (no user)
Externally authenticated user	AssumeRoleWithSAML/a	SAML user identity	SAML user identity	Role identity only (no user)
Externally authenticated user	AssumeRoleWithWebIdentity	OIDC/Web user identity	OIDC/Web user identity	Role identity only (no user)

Examples of Logged Events in CloudTrail Files

CloudTrail log files contain events that are formatted using JSON. An event represents a single API request or sign-in event and includes information about the requested action, any parameters, and the date and time of the action.

IAM API Event in CloudTrail Log File

The following example shows a CloudTrail log entry for a request made for the IAM `GetUserPolicy` action.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Alice",
    "accountId": "444455556666",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "userName": "Alice",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2014-07-15T21:39:40Z"
      }
    },
  },
}
```

```

        "invokedBy": "signin.amazonaws.com"
},
"eventTime": "2014-07-15T21:40:14Z",
"eventSource": "iam.amazonaws.com",
"eventName": " GetUserPolicy",
"awsRegion": "us-east-2",
"sourceIPAddress": "signin.amazonaws.com",
"userAgent": "signin.amazonaws.com",
"requestParameters": {
    "userName": "Alice",
    "policyName": "ReadOnlyAccess-Alice-201407151307"
},
"responseElements": null,
"requestID": "9EXAMPLE-0c68-11e4-a24e-d5e16EXAMPLE",
"eventID": "cEXAMPLE-127e-4632-980d-505a4EXAMPLE"
}

```

From this event information, you can determine that the request was made to get a user policy named `ReadOnlyAccess-Alice-201407151307` for user Alice, as specified in the `requestParameters` element. You can also see that the request was made by an IAM user named Alice on July 15, 2014 at 9:40 PM (UTC). In this case, the request originated in the AWS Management Console, as you can tell from the `userAgent` element.

AWS STS API Event in CloudTrail Log File

The IAM user named "Bob" in account 777788889999 calls the AWS STS AssumeRole action to assume the role EC2-dev in account 111122223333. The next two examples show the CloudTrail log entries for the two affected accounts. The first example shows the CloudTrail log entry for the request made in account 777788889999, the account that owns the user who calls `AssumeRole`.

```

{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAQRSTUVWXYZEXAMPLE",
        "arn": "arn:aws:iam::777788889999:user/Bob",
        "accountId": "777788889999",
        "accessKeyId": "AKIAQRSTUVWXYZEXAMPLE",
        "userName": "Bob"
    },
    "eventTime": "2014-07-18T15:07:39Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": " AssumeRole",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "192.0.2.101",
    "userAgent": "aws-cli/1.11.10 Python/2.7.8 Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 botocore/1.4.67",
    "requestParameters": {
        "roleArn": "arn:aws:iam::111122223333:role/EC2-dev",
        "roleSessionName": "Bob-EC2-dev",
        "serialNumber": "arn:aws:iam::777788889999:mfa"
    },
    "responseElements": {
        "credentials": {
            "sessionToken": "<encoded session token blob>",
            "accessKeyId": "AKIAQRSTUVWXYZEXAMPLE",
            "expiration": "Jul 18, 2014 4:07:39 PM"
        },
        "assumedRoleUser": {
            "assumedRoleId": "AIDAQRSTUVWXYZEXAMPLE:Bob-EC2-dev",
            "arn": "arn:aws:sts::111122223333:assumed-role/EC2-dev/Bob-EC2-dev"
        }
    }
}

```

```

"resources": [
  {
    "ARN": "arn:aws:iam::111122223333:role/EC2-dev",
    "accountId": "111122223333",
    "type": "AWS::IAM::Role"
  }
],
"requestID": "4EXAMPLE-0e8d-11e4-96e4-e55c0EXAMPLE",
"sharedEventID": "bEXAMPLE-efea-4a70-b951-19a88EXAMPLE",
"eventID": "dEXAMPLE-ac7f-466c-a608-4ac8dEXAMPLE"
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
}

```

The second example shows the role owning account's (111122223333) CloudTrail log entry for the same request.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AWSAccount",
    "principalId": "AIDAQRSTUVWXYZEXAMPLE",
    "accountId": "777788889999"
  },
  "eventTime": "2014-07-18T15:07:39Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRole",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/1.11.10 Python/2.7.8 Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 botocore/1.4.67",
  "requestParameters": {
    "roleArn": "arn:aws:iam:: 111122223333:role/EC2-dev",
    "roleSessionName": "Bob-EC2-dev",
    "serialNumber": "arn:aws:iam::777788889999:mfa"
  },
  "responseElements": {
    "credentials": {
      "sessionToken": "<encoded session token blob>",
      "accessKeyId": "AKIAQRSTUVWXYZEXAMPLE",
      "expiration": "Jul 18, 2014 4:07:39 PM"
    },
    "assumedRoleUser": {
      "assumedRoleId": "AIDAQRSTUVWXYZEXAMPLE:Bob-EC2-dev",
      "arn": "arn:aws:sts::111122223333:assumed-role/EC2-dev/Bob-EC2-dev"
    }
  },
  "requestID": "4EXAMPLE-0e8d-11e4-96e4-e55c0EXAMPLE",
  "sharedEventID": "bEXAMPLE-efea-4a70-b951-19a88EXAMPLE",
  "eventID": "dEXAMPLE-ac7f-466c-a608-4ac8dEXAMPLE"
}

```

The following example shows a CloudTrail log entry for a request made by an AWS service calling an API using permissions from an IAM role.

```

{
  "eventVersion": "1.04",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDAQRSTUVWXYZEXAMPLE:devdsk",
    "arn": "arn:aws:sts::777788889999:assumed-role/AssumeNothing/devdsk",
    "accountId": "777788889999",
    "accessKeyId": "AKIAQRSTUVWXYZEXAMPLE",
    "sessionToken": "AQC...="
  }
}

```

```

    "sessionContext": {
        "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2016-11-14T17:25:26Z"
        },
        "sessionIssuer": {
            "type": "Role",
            "principalId": "AIDAQRSTUVWXYZEXAMPLE",
            "arn": "arn:aws:iam::777788889999:role/AssumeNothing",
            "accountId": "777788889999",
            "userName": "AssumeNothing"
        }
    },
    "eventTime": "2016-11-14T17:25:45Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "DeleteBucket",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "[aws-cli/1.11.10 Python/2.7.8 Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 botocore/1.4.67]",
    "requestParameters": {
        "bucketName": "my-test-bucket-cross-account"
    },
    "responseElements": null,
    "requestID": "EXAMPLE463D56D4C",
    "eventID": "dEXAMPLE-265a-41e0-9352-4401bEXAMPLE",
    "eventType": "AwsApiCall",
    "recipientAccountId": "777788889999"
}

```

The following example shows a CloudTrail log entry for a request made for the AWS STS AssumeRoleWithSAML action.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "SAMLUser",
        "principalId": "<id of identity provider>:<canonical id of user>",
        "userName": "<canonical id of user>",
        "identityProvider": "<id of identity provider>"
    },
    "eventTime": "2016-03-23T01:39:57Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRoleWithSAML",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "192.0.2.101",
    "userAgent": "aws-cli/1.3.23 Python/2.7.6 Linux/2.6.18-164.el5",
    "requestParameters": {
        "SAMLAssertionID": "_c0046cEXAMPLEb9d4b8eEXAMPLE2619aEXAMPLE",
        "roleSessionName": "MyAssignedRoleSessionName",
        "durationSeconds": 3600,
        "roleArn": "arn:aws:iam::444455556666:role/SAMLTestRoleShibboleth",
        "principalArn": "arn:aws:iam::444455556666:saml-provider/Shibboleth"
    },
    "responseElements": {
        "subjectType": "transient",
        "issuer": "https://server.example.com/idp/shibboleth",
        "credentials": {
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "expiration": "Mar 23, 2016 2:39:57 AM",
            "sessionToken": "<encoded session token blob>"
        },
        "nameQualifier": "<id of identity provider>",
        "sessionToken": "<session token>"
    }
}
```

```

    "assumedRoleUser": {
        "assumedRoleId": "AROAD35QRSTUVWEXAMPLE:MyAssignedRoleSessionName",
        "arn": "arn:aws:sts::444455556666:assumed-role/SAMLTestRoleShibboleth/
MyAssignedRoleSessionName"
    },
    "subject": "<canonical id of user>",
    "audience": "https://signin.aws.amazon.com/saml"
},
"resources": [
{
    "ARN": "arn:aws:iam::444455556666:role/SAMLTestRoleShibboleth",
    "accountId": "444455556666",
    "type": "AWS::IAM::Role"
},
{
    "ARN": "arn:aws:iam::444455556666:saml-provider/test-saml-provider",
    "accountId": "444455556666",
    "type": "AWS::IAM::SAMLProvider"
}
],
"requestID": "6EXAMPLE-e595-11e5-b2c7-c974fEXAMPLE",
"eventID": "dEXAMPLE-265a-41e0-9352-4401bEXAMPLE",
"eventType": "AwsApiCall",
"recipientAccountId": "444455556666"
}
}

```

The following example shows a CloudTrail log entry for a request made for the AWS STS AssumeRoleWithWebIdentity action.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "WebIdentityUser",
        "principalId": "accounts.google.com:<id-of-application>.apps.googleusercontent.com:<id-
of-user>",
        "userName": "<id of user>",
        "identityProvider": "accounts.google.com"
    },
    "eventTime": "2016-03-23T01:39:51Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRoleWithWebIdentity",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "192.0.2.101",
    "userAgent": "aws-cli/1.3.23 Python/2.7.6 Linux/2.6.18-164.el5",
    "requestParameters": {
        "durationSeconds": 3600,
        "roleArn": "arn:aws:iam::444455556666:role/FederatedWebIdentityRole",
        "roleSessionName": "MyAssignedRoleSessionName"
    },
    "responseElements": {
        "provider": "accounts.google.com",
        "subjectFromWebIdentityToken": "<id of user>",
        "audience": "<id of application>.apps.googleusercontent.com",
        "credentials": {
            "accessKeyId": "ASIAQQRSTUVWRAOEXAMPLE",
            "expiration": "Mar 23, 2016 2:39:51 AM",
            "sessionToken": "<encoded session token blob>"
        },
        "assumedRoleUser": {
            "assumedRoleId": "AROACQRSTUVWRAOEXAMPLE:MyAssignedRoleSessionName",
            "arn": "arn:aws:sts::444455556666:assumed-role/FederatedWebIdentityRole/
MyAssignedRoleSessionName"
        }
    }
}
```

```

"resources": [
  {
    "ARN": "arn:aws:iam::444455556666:role/FederatedWebIdentityRole",
    "accountId": "444455556666",
    "type": "AWS::IAM::Role"
  }
],
"requestID": "6EXAMPLE-e595-11e5-b2c7-c974fEXAMPLE",
"eventID": "bEXAMPLE-0b30-4246-b28c-e3da3EXAMPLE",
"eventType": "AwsApiCall",
"recipientAccountId": "444455556666"
}

```

Sign-in Failure Event in CloudTrail Log File

The following example shows a CloudTrail log entry for a failed sign-in event.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "userName": "Alice"
  },
  "eventTime": "2014-07-08T17:35:27Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "ConsoleLogin",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.100",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0",
  "errorMessage": "Failed authentication",
  "requestParameters": null,
  "responseElements": {
    "ConsoleLogin": "Failure"
  },
  "additionalEventData": {
    "MobileVersion": "No",
    "LoginTo": "https://console.aws.amazon.com/sns",
    "MFAUsed": "No"
  },
  "eventID": "11ea990b-4678-4bcd-8fbe-62509088b7cf"
}

```

From this information, you can determine that the sign-in attempt was made by an IAM user named Alice, as shown in the `userIdentity` element. You can also see that the sign-in attempt failed, as shown in the `responseElements` element. You can see that Alice tried to sign in to the Amazon SNS console at 5:35 PM (UTC) on July 8th, 2014.

Sign-in Failure Event Caused by Incorrect User Name

The following example shows a CloudTrail log entry for an unsuccessful sign-in event caused by the user entering an incorrect user name. AWS masks the `userName` text with `HIDDEN_DUE_TO_SECURITY_REASON` to help prevent exposing potentially sensitive information.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "accountId": "123456789012",
    "accessKeyId": ""
  }
}

```

```
        "userName": "HIDDEN_DUE_TO_SECURITY_REASON$"
    },
    "eventTime": "2015-03-31T22:20:42Z",
    "eventSource": "signin.amazonaws.com",
    "eventName": "ConsoleLogin",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "192.0.2.101",
    "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0",
    "errorMessage": "No username found in supplied account",
    "requestParameters": null,
    "responseElements": {
        "ConsoleLogin": "Failure"
    },
    "additionalEventData": {
        "LoginTo": "https://console.aws.amazon.com/console/home?state=hashArgs%23&isauthcode=true",
        "MobileVersion": "No",
        "MFAUsed": "No"
    },
    "eventID": "a7654656-0417-45c6-9386-ea8231385051",
    "eventType": "AwsConsoleSignin",
    "recipientAccountId": "123456789012"
}
```

Sign-in Success Event in CloudTrail Log File

The following example shows a CloudTrail log entry for a successful sign-in event.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/Bob",
        "accountId": "111122223333",
        "userName": "Bob"
    },
    "eventTime": "2014-07-16T15:49:27Z",
    "eventSource": "signin.amazonaws.com",
    "eventName": "ConsoleLogin",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "192.0.2.110",
    "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0",
    "requestParameters": null,
    "responseElements": {
        "ConsoleLogin": "Success"
    },
    "additionalEventData": {
        "MobileVersion": "No",
        "LoginTo": "https://console.aws.amazon.com/s3",
        "MFAUsed": "No"
    },
    "eventID": "3fcfb182-98f8-4744-bd45-10a395ab61cb"
}
```

For more details about the information contained in CloudTrail log files, see [CloudTrail Event Reference](#) in the [AWS CloudTrail User Guide](#).

Preventing Duplicate Log Entries in CloudTrail

CloudTrail creates trails in each region separately. These trails include information for events that occur in those regions, plus global (that is, non-region-specific) events such as IAM API calls, non-region

specific AWS STS calls (calls to `sts.amazonaws.com`), and AWS sign-in events. For example, by default, if you have two trails, each in a different region, and you then create a new IAM user, the `CreateUser` event is added to the log files in both regions, creating a duplicate log entry.

Note

By default, AWS Security Token Service (STS) is a global service with a single endpoint at <https://sts.amazonaws.com>. Calls to this endpoint are logged as calls to a global service. However, because this endpoint is physically located in the US East (N. Virginia) region, your logs list "us-east-1" as the event region. CloudTrail does not write these logs to the US East (Ohio) region unless you choose to include global service logs in that region. CloudTrail writes calls to all regional endpoints to their respective regions. For example, calls to `sts.us-east-2.amazonaws.com` are published to the US East (Ohio) region, calls to `sts.eu-central-1.amazonaws.com` are published to the EU (Frankfurt) region, etc. For more information about multiple regions and AWS STS see [Activating and Deactivating AWS STS in an AWS Region \(p. 257\)](#).

The following table lists the regions and how CloudTrail logs AWS STS requests in each region. The "Location" column indicates which logs CloudTrail writes to. "Global" means the event is logged in any region for which you choose to include global service logs in that region. "Region" means that the event is logged only in the region where the endpoint is located. The last column indicates how the request's region is identified in the log entry.

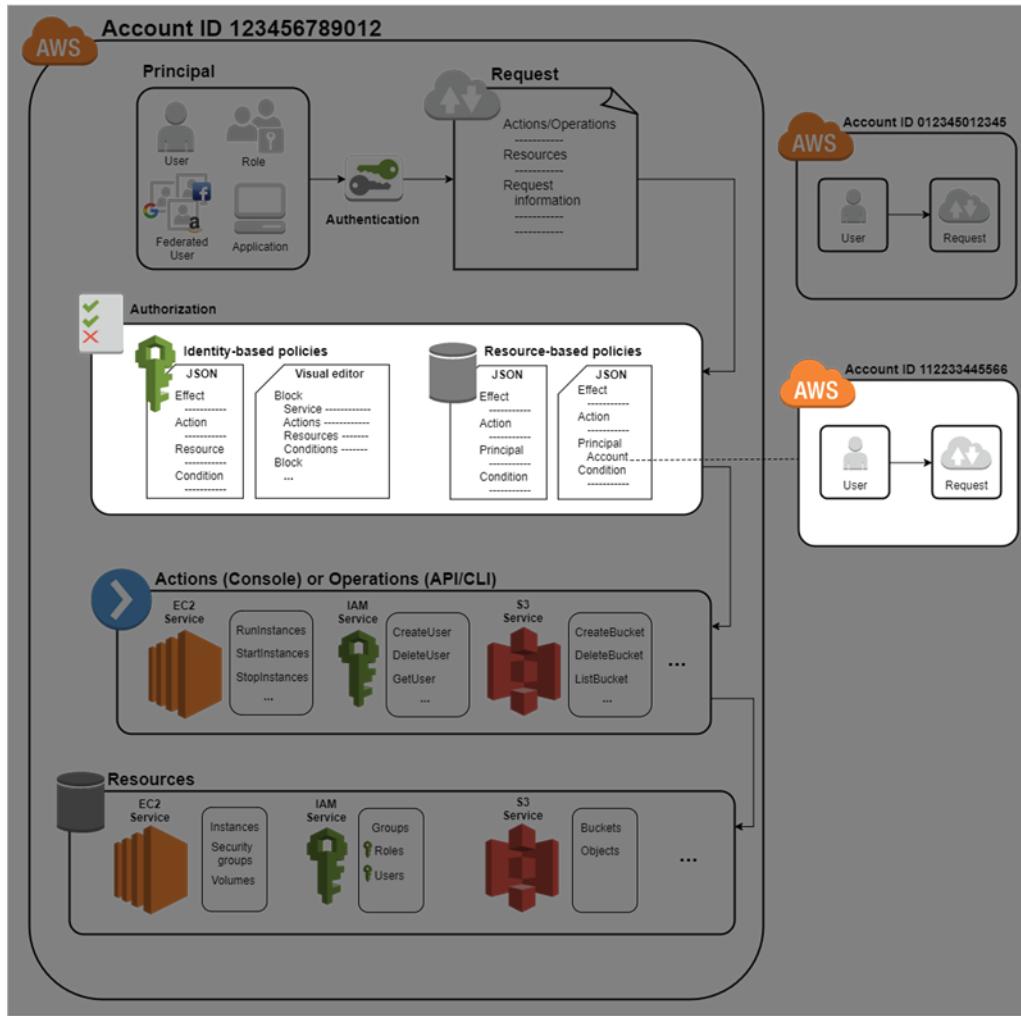
Region name	Region identity in CloudTrail log	Endpoint	Location of CloudTrail logs
n/a - global	us-east-1	<code>sts.amazonaws.com</code>	global
US East (Ohio)	us-east-2	<code>sts.us-east-2.amazonaws.com</code>	region
US East (N. Virginia)	us-east-1	<code>sts.us-east-1.amazonaws.com</code>	region
US West (N. California)	us-west-1	<code>sts.us-west-1.amazonaws.com</code>	region
US West (Oregon)	us-west-2	<code>sts.us-west-2.amazonaws.com</code>	region
Canada (Central)	ca-central-1	<code>sts.ca-central-1.amazonaws.com</code>	region
EU (Frankfurt)	eu-central-1	<code>sts.eu-central-1.amazonaws.com</code>	region
EU (Ireland)	eu-west-1	<code>sts.eu-west-1.amazonaws.com</code>	region
EU (London)	eu-west-2	<code>sts.eu-west-2.amazonaws.com</code>	region
Asia Pacific (Tokyo)	ap-northeast-1	<code>sts.ap-northeast-1.amazonaws.com</code>	region
Asia Pacific (Seoul)	ap-northeast-2	<code>sts.ap-northeast-2.amazonaws.com</code>	region
Asia Pacific (Mumbai)	ap-south-1	<code>sts.ap-south-1.amazonaws.com</code>	region
Asia Pacific (Singapore)	ap-southeast-1	<code>sts.ap-southeast-1.amazonaws.com</code>	region
Asia Pacific (Sydney)	ap-southeast-2	<code>sts.ap-southeast-2.amazonaws.com</code>	region
South America (São Paulo)	sa-east-1	<code>sts.sa-east-1.amazonaws.com</code>	region

When you configure CloudTrail to aggregate trail information from multiple regions in your account into a single Amazon S3 bucket, IAM events are duplicated in the logs—the trail for each region writes the same IAM event to the aggregated log. To prevent this duplication, you can include global events selectively. A typical approach is to enable global events in one trail and to disable global events in all other trails that write to the same Amazon S3 bucket. That way only one set of global events is written.

For more information, see [Aggregating Logs](#) in the *AWS CloudTrail User Guide*.

Access Management

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. When a [principal \(p. 4\)](#) makes a request in AWS, the IAM service checks whether the principal is authenticated (signed in) and authorized (has permissions). You manage access by creating policies and attaching them to IAM identities or AWS resources. Those policies specify the permissions that are allowed or denied. For details about the rest of the authentication and authorization process, see [Understanding How IAM Works \(p. 3\)](#).



During authorization, IAM uses values from the [request context \(p. 4\)](#) to check for matching policies and determine whether to allow or deny the request.

Policies are stored in AWS as JSON documents and specify the permissions that are allowed or denied for principals (*identity-based policies*) or resources (*resource-based policies*). Identity-based policies include AWS managed policies, customer managed policies, and inline policies. Resource-based policies also include trust policies. For more information about these types of policies, see [IAM Policies \(p. 275\)](#).

IAM checks each policy that matches the context of the request. If a single policy includes a denied action, IAM denies the entire request and stops evaluating policies. This is called an *explicit deny*. Because requests are *denied by default*, IAM authorizes your request only if every part of your request is allowed by the matching policies. The [evaluation logic \(p. 458\)](#) follows these rules:

- By default, all requests are denied.
- An explicit allow overrides this default.
- An explicit deny overrides any allows.

Note

By default, only the AWS account root user has access to all of the resources in that account. So if you are not signed in as the root user, you must have permissions granted by a policy.

After your request has been authenticated and authorized, AWS approves the request. If you need to make a request in a different account, the resource in that account must have a resource-based policy that allows access from your account. Otherwise, you must assume a role within that account with the permissions that you need.

Access Management Resources

For more information about permissions and about creating policies, see the following resources:

- The following entries in the AWS Security Blog cover common ways to write policies for access to Amazon S3 buckets and objects.
 - [Writing IAM Policies: How to Grant Access to an Amazon S3 Bucket](#)
 - [Writing IAM policies: Grant Access to User-Specific Folders in an Amazon S3 Bucket](#)
 - [IAM Policies and Bucket Policies and ACLs! Oh My! \(Controlling Access to S3 Resources\)](#)
 - [A Primer on RDS Resource-Level Permissions](#)
 - [Demystifying EC2 Resource-Level Permissions](#)

IAM Policies

A policy is an entity in AWS that, when attached to an identity or resource, defines their permissions. AWS evaluates these policies when a principal, such as a user, makes a request. Permissions in the policies determine whether the request is allowed or denied. Policies are stored in AWS as JSON documents attached to principals as *identity-based policies*, or to resources as *resource-based policies*.

Identity-Based Policies

Identity-based policies are permission policies that you can attach to a principal (or identity), such as an IAM user, role, or group. These policies control what actions that identity can perform, on which resources, and under what conditions. Identity-based policies can be further categorized:

- **Managed policies** – Standalone identity-based policies that you can attach to multiple users, groups, and roles in your AWS account. You can use two types of managed policies:
 - **AWS managed policies** – Managed policies that are created and managed by AWS. If you are new to using policies, we recommend that you start by using AWS managed policies.
 - **Customer managed policies** – Managed policies that you create and manage in your AWS account. Customer managed policies provide more precise control over your policies than AWS managed policies. You can create and edit an IAM policy in the visual editor or by creating the JSON policy document directly. For more information, see [Creating IAM Policies \(p. 317\)](#) and [Editing IAM Policies \(p. 337\)](#).
- **Inline policies** – Policies that you create and manage and that are *embedded* directly into a single user, group, or role.

To learn how to choose whether to use a managed policy or an inline policy, see [Managed Policies and Inline Policies \(p. 279\)](#).

Resource-Based Policies

Resource-based policies are JSON policy documents that you attach to a resource such as an Amazon S3 bucket. These policies control what actions a specified principal can perform on that resource and under what conditions. Resource-based policies are inline policies, and there are no managed resource-based policies.

Although IAM identities are technically AWS resources, you cannot attach a resource-based policy to an IAM identity. You must use identity-based policies in IAM. To see which services support resource-based policies, see [AWS Services That Work with IAM \(p. 417\)](#). To learn more about resource-based policies, see [Identity-Based Policies and Resource-Based Policies \(p. 285\)](#).

Trust policies are resource-based policies that are attached to a role that define which principals can assume the role. When you create a role in IAM, the role must have two things: The first is a trust policy that indicates who can assume the role. The second is a permission policy that indicates what they can do with that role. Remember that adding an account to the trust policy of a role is only half of establishing the trust relationship. By default, no users in the trusted accounts can assume the role until the administrator for that account grants the users the permission to assume the role. For more information, see [Granting a User Permissions to Switch Roles \(p. 205\)](#).

Important

Throughout the AWS documentation, when we refer to an IAM policy without mentioning any of the specific categories above, we mean an identity-based, customer managed policy.

Overview of JSON Policies

Policies are stored in AWS as JSON documents attached to principals as *identity-based policies*, or to resources as *resource-based policies*. It is not necessary for you to understand the JSON syntax. You can use the visual editor to create and edit customer managed policies without ever using JSON. However, if you choose to use inline policies for groups, you are still required to create and edit those policies in the JSON editor. For more information about using the visual editor, see [Creating IAM Policies \(p. 317\)](#) and [Editing IAM Policies \(p. 337\)](#).

Introduction to JSON Policies

To assign permissions to a user, group, role, or resource, you create a JSON *policy*, which is a document that defines permissions. The policy document includes the following elements:

- **Effect** – whether the policy allows or denies access
- **Action** – the list of actions that are allowed or denied by the policy
- **Resource** – the list of resources on which the actions can occur
- **Condition (Optional)** – the circumstances under which the policy grants permission

To learn about these and other policy elements, see [IAM JSON Policy Elements Reference \(p. 426\)](#).

Policies are documents that are stored using JSON. A policy consists of one or more *statements*, each of which describes one set of permissions. Here's an example of a simple policy.

```
{  
  "Version": "2012-10-17",  
  "Statement": {  
    "Effect": "Allow",
```

```
    "Action": "s3>ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
}
```

You can attach this policy to an IAM identity (group, user, or role). If that's the only policy for the identity, the identity is allowed to perform only this one action (`ListBucket`) on one Amazon S3 bucket (`example_bucket`).

To specify permissions for a resource, you can attach a resource-based policy to the resource, such as an Amazon SNS topic, an Amazon S3 bucket, or an Amazon Glacier vault. In that case, the policy has to include information about who is allowed to access the resource, known as the *principal*. (For identity-based policies, the principal is the IAM user that the policy is attached to, or the user who gets the policy from a group.) For more information, see [Identity-Based Policies and Resource-Based Policies \(p. 285\)](#).

The following example shows a resource-based policy that might be attached to an Amazon S3 bucket. The policy grants permission to a specific AWS account to perform any Amazon S3 actions in `mybucket`. This includes both working with the bucket and with the objects in it. (Because the policy grants trust only to the account, individual users in the account must still be granted permissions for the specified Amazon S3 actions.)

```
{
  "Version": "2012-10-17",
  "Id": "S3-Account-Permissions",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": {"AWS": ["arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:root"]},
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::mybucket",
        "arn:aws:s3:::mybucket/*"
      ]
    }
  ]
}
```

IAM policies control access regardless of the interface. For example, you could provide a user with a password to access the AWS Management Console. The policies for that user (or any groups the user belongs to) would control what the user can do in the AWS Management Console. Or, you could provide the user with AWS access keys for making API calls to AWS. The policies would control what actions the user could call through a library or client that uses those access keys for authentication.

For basic example policies that cover common scenarios, see [Example Policies \(p. 295\)](#).

AWS managed policies and the Policy Generator are available from the IAM console in the AWS Management Console. For more information about creating policies in the console, see [Managing IAM Policies \(p. 316\)](#). Also, you can use the [AWS Policy Generator](#) online to create policies for AWS products without accessing the console.

Multiple Statements and Multiple Policies

You can attach more than one policy to an entity. If you have multiple permissions to grant to an entity, you can put them in separate policies, or you can put them all in one policy.

Generally, each statement in a policy includes information about a single permission. If your policy includes multiple statements, a logical OR is applied across the statements at evaluation time. Similarly, if multiple policies are applicable to a request, a logical OR is applied across the policies at evaluation time.

Users often have multiple policies that apply to them (but aren't necessarily *attached* to them). For example, IAM user Bob could have policies attached to him, and other policies attached to the groups

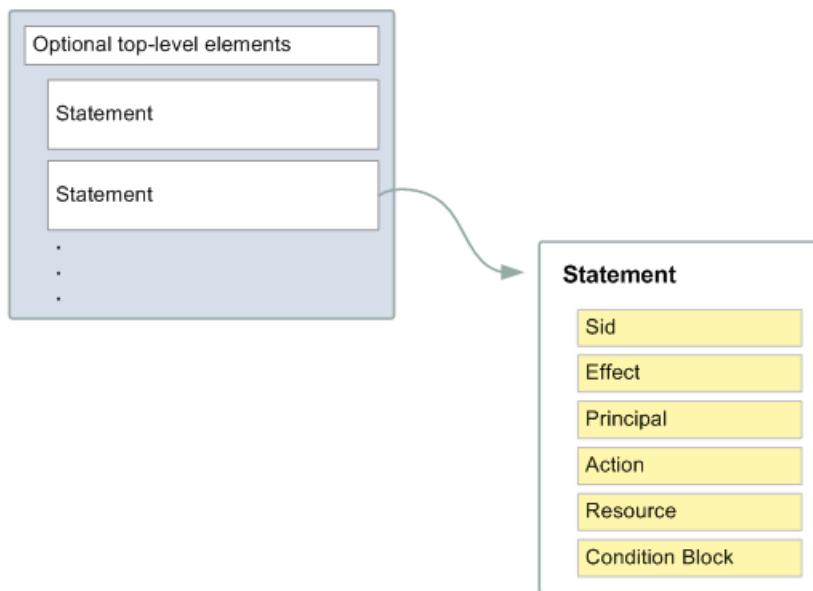
he's in. In addition, he might be accessing an Amazon S3 bucket that has its own bucket policy (resource-based policy). All applicable policies are evaluated and the result is always that access is either granted or denied. For more information about the evaluation logic we use, see [IAM JSON Policy Evaluation Logic \(p. 458\)](#).

Policy Structure

Each policy is a JSON document. As illustrated in the following figure, a policy includes:

- Optional policy-wide information (at the top of the document)
- One or more individual *statements*

Each statement includes the core information about a single permission. If a policy includes multiple statements, AWS applies a logical OR across the statements at evaluation time. If multiple policies are applicable to a request, AWS applies a logical OR across the policies at evaluation time.



The information in a statement is contained within a series of *elements*. For information about these elements, see [IAM JSON Policy Elements Reference \(p. 426\)](#).

Example Policy with Multiple Statements

Policies often include multiple statements, where each statement grants permissions to a different set of resources or grants permissions under a specific condition. For example, the following policy has three statements, each of which grants a separate set of permissions. Assume that the user or group that the policy is attached to is in AWS account 123456789012. (Policies can't reference resources in other accounts.) The statements do the following:

- The first statement, with a `Sid` (Statement ID) element set to `FirstStatement`, lets users manage their own passwords. The `Resource` element in this statement is `"*"` (which means "all resources"), but in practice, the `ChangePassword` API (or equivalent `change-password` CLI command) affects only the password for the user who makes the request.
- The second statement (`"Sid": "SecondStatement"`) lets the user list all the Amazon S3 buckets in their AWS account. The `Resource` element in this statement is `"*"` (which means "all resources"). But because policies don't grant access to resources in other accounts, the user can list only the buckets

in their own AWS account. (This permission is necessary for the user to access a bucket from the AWS Management Console.)

- The third statement ("Sid": "ThirdStatement") lets the user list and retrieve any object that is in a bucket called confidential-data, but only when the user is authenticated with short-term credentials validated by a multi-factor authentication (MFA) device. The Condition element in the policy checks whether the user is MFA-authenticated, and if so, the user can list and retrieve objects in the bucket.

When a policy statement contains a Condition element, the statement is only in effect when the Condition element evaluates to true. In this case, the Condition evaluates to true when the user is MFA-authenticated. If the user is not MFA-authenticated, this Condition evaluates to false. In that case, the third statement in this policy will not take effect, so the user will not have access to the confidential-data bucket.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "FirstStatement",  
            "Effect": "Allow",  
            "Action": ["iam:ChangePassword"],  
            "Resource": "*"  
        },  
        {  
            "Sid": "SecondStatement",  
            "Effect": "Allow",  
            "Action": "s3>ListAllMyBuckets",  
            "Resource": "*"  
        },  
        {  
            "Sid": "ThirdStatement",  
            "Effect": "Allow",  
            "Action": [  
                "s3>List*",  
                "s3:Get*"  
            ],  
            "Resource": [  
                "arn:aws:s3:::confidential-data",  
                "arn:aws:s3:::confidential-data/*"  
            ],  
            "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}  
        }  
    ]  
}
```

Managed Policies and Inline Policies

When you need to set the permissions for an identity in IAM, you must decide whether to use an AWS managed policy, a customer managed policy, or an inline policy. The following sections provide more information about each of the types of identity-based policies and when to use them.

Topics

- [AWS Managed Policies \(p. 280\)](#)
- [Customer Managed Policies \(p. 281\)](#)
- [Inline Policies \(p. 282\)](#)
- [Choosing Between Managed Policies and Inline Policies \(p. 283\)](#)
- [Deprecated AWS Managed Policies \(p. 284\)](#)

AWS Managed Policies

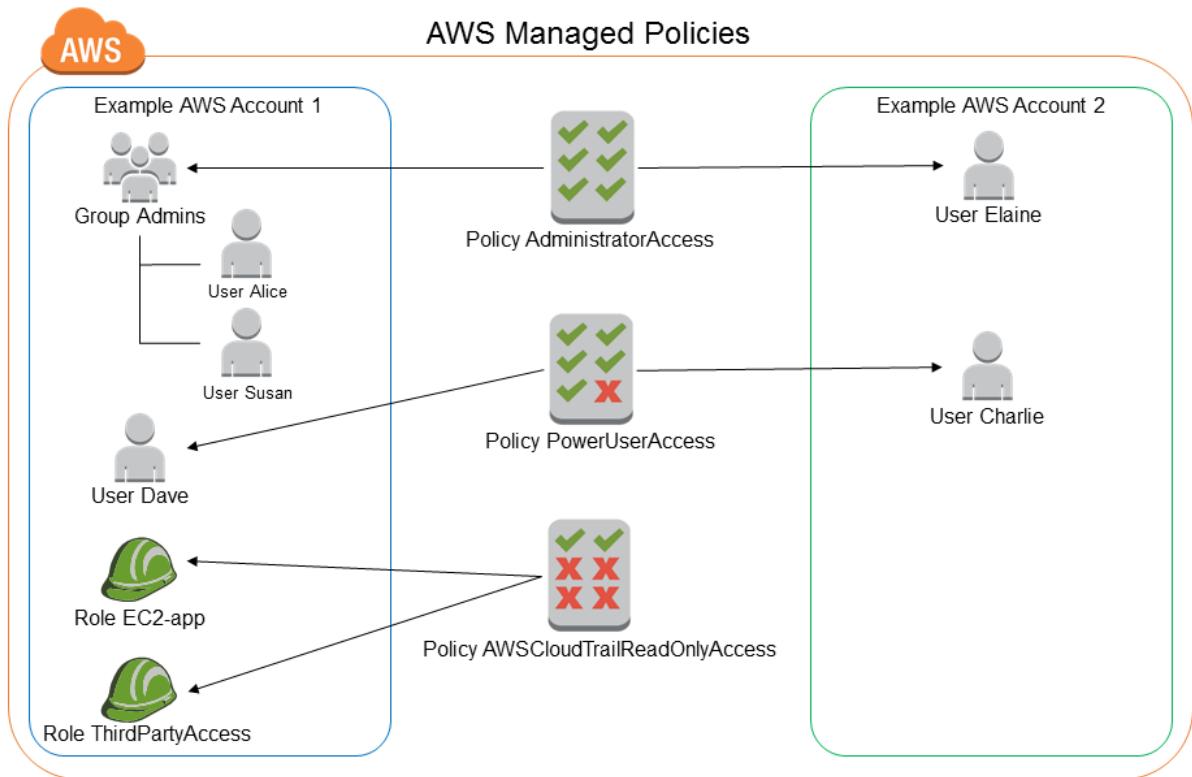
An *AWS managed policy* is a standalone policy that is created and administered by AWS. *Standalone policy* means that the policy has its own [Amazon Resource Name \(ARN\)](#) that includes the policy name.

AWS managed policies are designed to provide permissions for many common use cases. There are AWS managed policies that define typical permissions for service administrators and grant full access to the service, such as [AmazonDynamoDBFullAccess](#) and [IAMFullAccess](#). Some AWS managed policies are designed for power users, such as [AWSCodeCommitPowerUser](#) and [AWSKeyManagementServicePowerUser](#), which also depends on IAM permissions. Other AWS managed policies provide various levels of access to AWS services, such as [AmazonMobileAnalyticsWriteOnlyAccess](#) and [AmazonEC2ReadOnlyAccess](#), which provides access to additional related services. AWS managed policies make it easier for you to assign appropriate permissions to users, groups, and roles than if you had to write the policies yourself.

One particularly useful category of AWS managed policies are those designed for job functions. These policies align closely to commonly used job functions in the IT industry. The intent is to make granting permissions for these common job functions easy. One key advantage of using job function policies is that they are maintained and updated by AWS as new services and API operations are introduced. For example, the [AdministratorAccess](#) job function provides full access and permissions delegation to every service and resource in AWS. We recommend that this policy is used only for the account administrator. For power users that require full access to every service except limited access to IAM and Organizations, use the [PowerUserAccess](#) job function. For a list and descriptions of the job function policies, see [AWS Managed Policies for Job Functions \(p. 468\)](#).

You cannot change the permissions defined in AWS managed policies. AWS will occasionally update the permissions defined in an AWS managed policy. When AWS does this, the update affects all principal entities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API calls become available for existing services. For example, the AWS managed policy called [ReadOnlyAccess](#) provides read-only access to all AWS services and resources. When AWS launches a new service, AWS updates the [ReadOnlyAccess](#) policy to add read-only permissions for the new service. The updated permissions are applied to all principal entities that the policy is attached to.

The following diagram illustrates AWS managed policies. The diagram shows three AWS managed policies: [AdministratorAccess](#), [PowerUserAccess](#), and [AWSCloudTrailReadOnlyAccess](#). Notice that a single AWS managed policy can be attached to principal entities in different AWS accounts, and to different principal entities in a single AWS account.



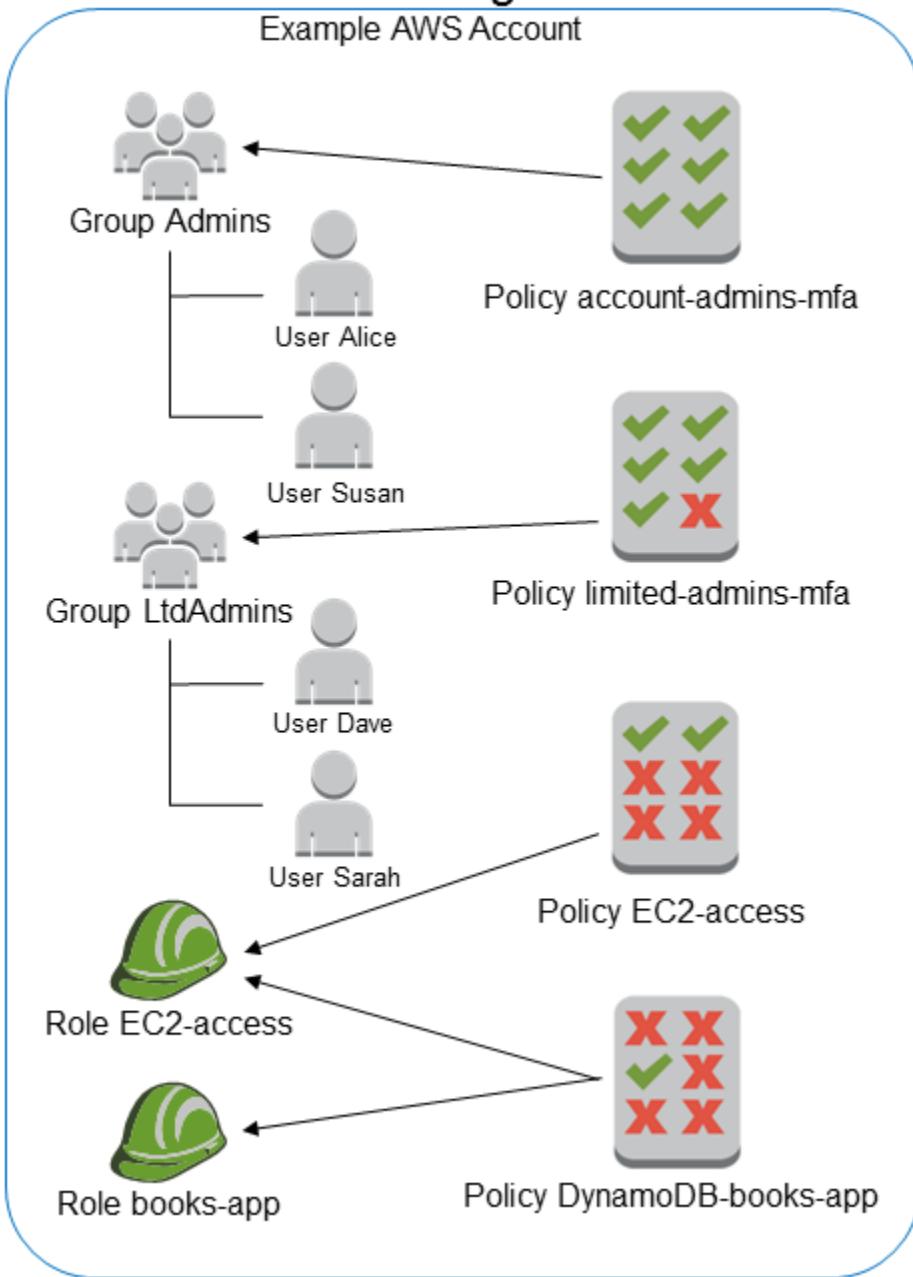
Customer Managed Policies

You can create standalone policies that you administer in your own AWS account, which we refer to as *customer managed policies*. You can then attach the policies to multiple principal entities in your AWS account. When you attach a policy to a principal entity, you give the entity the permissions that are defined in the policy.

A great way to create a customer managed policy is to start by copying an existing AWS managed policy. That way you know that the policy is correct at the beginning and all you need to do is customize it to your environment.

The following diagram illustrates customer managed policies. Each policy is an entity in IAM with its own [Amazon Resource Name \(ARN\)](#) that includes the policy name. Notice that the same policy can be attached to multiple principal entities—for example, the same **DynamoDB-books-app** policy is attached to two different IAM roles.

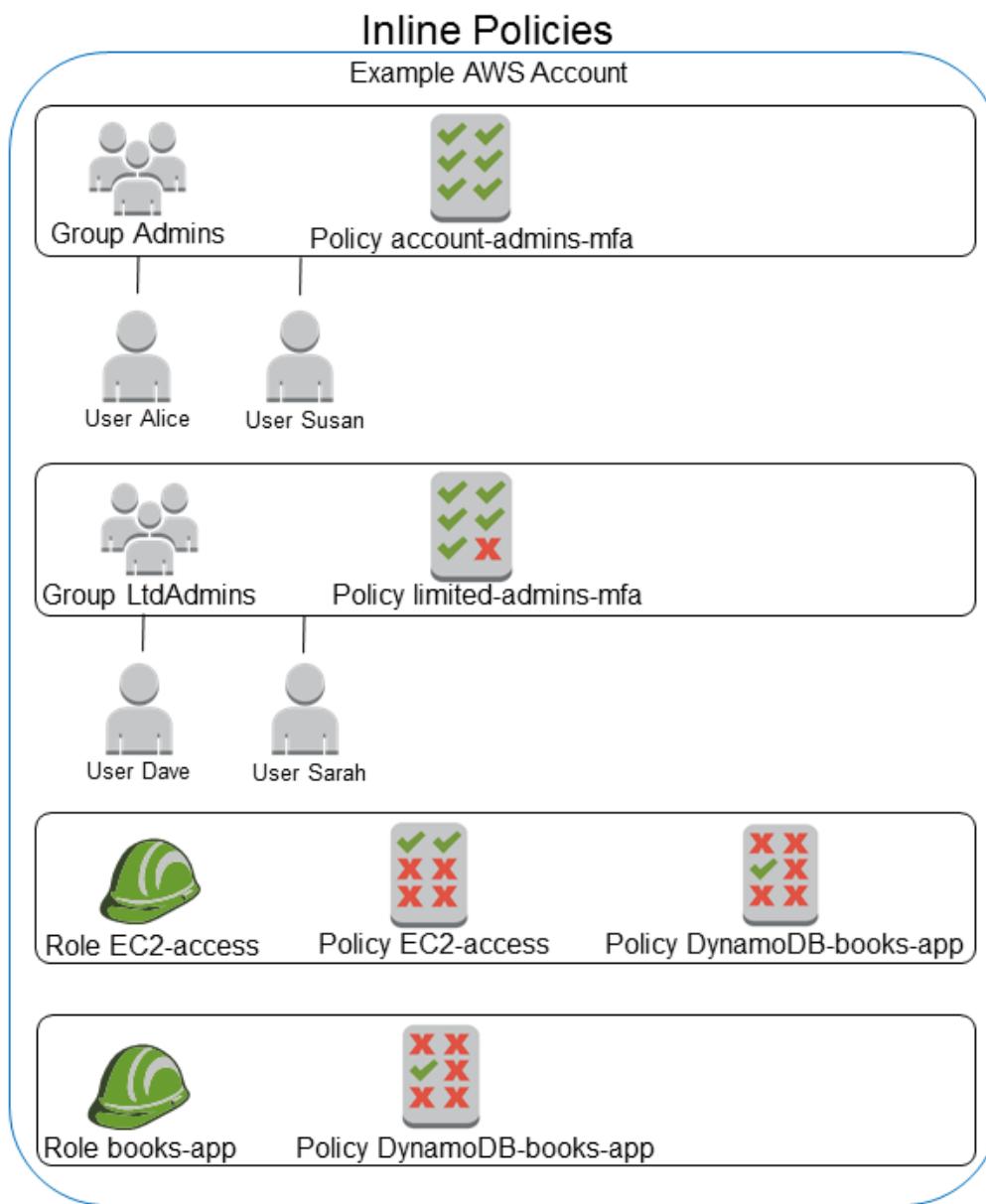
Customer Managed Policies



Inline Policies

An inline policy is a policy that's embedded in a principal entity (a user, group, or role)—that is, the policy is an inherent part of the principal entity. You can create a policy and embed it in a principal entity, either when you create the principal entity or later.

The following diagram illustrates inline policies. Each policy is an inherent part of the user, group, or role. Notice that two roles include the same policy (the **DynamoDB-books-app** policy), but they are not sharing a single policy; each role has its own copy of the policy.



Choosing Between Managed Policies and Inline Policies

The different types of policies are for different use cases. In most cases, we recommend that you use managed policies instead of inline policies.

Managed policies provide the following features:

Reusability

A single managed policy can be attached to multiple principal entities (users, groups, and roles). In effect, you can create a library of policies that define permissions that are useful for your AWS account, and then attach these policies to principal entities as needed.

Central change management

When you change a managed policy, the change is applied to all principal entities that the policy is attached to. For example, if you want to add permission for a new AWS API, you can update the managed policy to add the permission. (If you're using an AWS managed policy, AWS updates to the policy.) When the policy is updated, the changes are applied to all principal entities that the policy is attached to. In contrast, to change an inline policy you must individually edit each principal entity that contains the policy. For example, if a group and a role both contain the same inline policy, you must individually edit both principal entities in order to change that policy.

Versioning and rolling back

When you change a customer managed policy, the changed policy doesn't overwrite the existing policy. Instead, IAM creates a new version of the managed policy. IAM stores up to five versions of your customer managed policies. You can use policy versions to revert a policy to an earlier version if you need to.

A policy version is different from a `Version` policy element. The `Version` policy element is used within a policy and defines the version of the policy language. To learn more about policy versions, see [the section called "Versioning IAM Policies" \(p. 335\)](#). To learn more about the `Version` policy element see [IAM JSON Policy Elements: Version \(p. 426\)](#).

Delegating permissions management

You can allow users in your AWS account to attach and detach policies while maintaining control over the permissions defined in those policies. In effect, you can designate some users as full admins—that is, admins that can create, update, and delete policies. You can then designate other users as limited admins. That is, admins that can attach policies to other principal entities, but only the policies that you have allowed them to attach.

For more information about delegating permissions management, see [Controlling Access to Policies \(p. 291\)](#).

Automatic updates for AWS managed policies

AWS maintains AWS managed policies and updates them when necessary (for example, to add permissions for new AWS services), without you having to make changes. The updates are automatically applied to the principal entities that you have attached the AWS managed policy to.

Using Inline Policies

Inline policies are useful if you want to maintain a strict one-to-one relationship between a policy and the principal entity that it's applied to. For example, you want to be sure that the permissions in a policy are not inadvertently assigned to a principal entity other than the one they're intended for. When you use an inline policy, the permissions in the policy cannot be inadvertently attached to the wrong principal entity. In addition, when you use the AWS Management Console to delete that principal entity, the policies embedded in the principal entity are deleted as well. That's because they are part of the principal entity.

Deprecated AWS Managed Policies

To simplify the assignment of permissions, AWS provides [managed policies \(p. 279\)](#)—predefined policies that are ready to be attached to your IAM users, groups, and roles.

Sometimes AWS needs to add a new permission to an existing policy, such as when a new service is introduced. Adding a new permission to an existing policy does not disrupt or remove any feature or ability.

However, AWS might choose to create a *new* policy when the needed changes could impact customers if they were applied to an existing policy. For example, removing permissions from an existing policy could

break the permissions of any IAM entity or application that depended upon it, potentially disrupting a critical operation.

Therefore, when such a change is required, AWS creates a completely new policy with the required changes and makes it available to customers. The old policy is then marked *deprecated*. A deprecated managed policy appears with a warning icon next to it in the **Policies** list in the IAM console.

A deprecated policy has the following characteristics:

- It continues to work for all *currently* attached users, groups, and roles. Nothing breaks.
- It *cannot* be attached to any new users, groups, or roles. If you detach it from a current entity, you cannot reattach it.
- After you detach it from all current entities, it is no longer visible and can no longer be used in any way.

If any user, group, or role requires the policy, you must instead attach the new policy. When you receive notice that a policy is deprecated, we recommend that you immediately plan to attach all users, groups, and roles to the replacement policy and detach them from the deprecated policy. Continuing to use the deprecated policy can carry risks that are mitigated only by switching to the replacement policy.

Identity-Based Policies and Resource-Based Policies

When you create a policy to restrict access to a resource, you can choose an *identity-based policy* or a *resource-based policy*.

- Identity-based IAM policies are attached to an IAM user, group, or role. These policies let you specify what that user, group, or role can do. For example, you can attach the policy to the IAM user named Bob, stating that he has permission to use the Amazon Elastic Compute Cloud (Amazon EC2) RunInstances action. The policy could further state that Bob has permission to get items from an Amazon DynamoDB table named MyCompany. You can also grant Bob access to manage his own IAM security credentials. Identity-based permissions can be [managed or inline](#) (p. 279).
- Resource-based policies are attached to a resource. For example, you can attach resource-based policies to Amazon S3 buckets, Amazon SQS queues, and AWS Key Management Service encryption keys. For a list of services that support resource-based permissions, see [AWS Services That Work with IAM](#) (p. 417). With resource-based policies you can specify who has access to the resource and what actions they can perform on it. Resource-based policies are inline only, not managed.

Note

Resource-based policies differ from *resource-level* permissions. You can attach resource-based policies directly to a resource, as described in this topic. Resource-level permissions refer to the ability to use ARNs to specify individual resources in a policy. Resource-based permissions are supported only by some AWS services. For a list of which services support resource-based policies and resource-level permissions, see [AWS Services That Work with IAM](#) (p. 417).

The following figure illustrates both types of policies. The first column shows policies attached to identities (two users and two groups). Some of those policies identify specific resources that the actions can be used against. Those actions support *resource-level* permissions. The second column shows policies attached to resources. Those services support *resource-based* permissions within *identity-based* policies.



Note

When you create a policy, AWS validates, processes, and transforms the policy before storing it. When AWS returns the policy in response to a user query or displays it in the console, AWS transforms the policy back into a human-readable format. This can result in differences in what you see in the policy visual editor: Visual editor permission blocks can be added, removed, or reordered, and content within a block can be optimized. In the policy JSON editor, insignificant white space can be removed, and elements within JSON maps can be reordered. In addition, AWS account IDs within the principal elements can be replaced by the ARN of the AWS account root user. Because of these possible changes, you should not compare JSON policy documents as strings.

A user who has specific permissions might request a resource that also has a permissions policy attached to it. In that case, AWS evaluates both sets of permissions when determining whether to grant access to the resource. For information about how policies are evaluated, see [IAM JSON Policy Evaluation Logic \(p. 458\)](#).

Note

Amazon S3 supports policies both for IAM users and for resources (referred to in Amazon S3 as *bucket policies*). In addition, Amazon S3 supports a permission mechanism known as an *access control list (ACL)* that is independent of IAM policies and permissions. You can use IAM policies in combination with Amazon S3 ACLs. For more information, see [Access Control](#) in the *Amazon Simple Storage Service Developer Guide*.

Controlling Access Using Policies

You can use a policy to control access to resources within IAM or all of AWS.

To use a [policy \(p. 275\)](#) to control access in AWS, you must understand how IAM grants access. AWS is composed of collections of *resources*. An IAM user is a resource. An Amazon S3 bucket is a resource. When you use the AWS API, the AWS CLI, or the AWS Management Console to take an action (such as creating a user), you send a *request* for that action. Your request specifies an action, a resource, a *principal* (group, user, or role), a *principal account*, and any necessary request information. All of this information provides *context*.

IAM then checks that you (the principal) are authenticated (signed in) and authorized (have permission) to perform the specified action on the specified resource. During authorization, IAM checks all the policies attached to your user or role and the policies attached to the resource that you are trying to access. These checks are based on the context of your request. AWS authorizes the request only if each part of your request is allowed by the policies.

To view a diagram of this process, see [Understanding How IAM Works \(p. 3\)](#). For details about how IAM determines whether a request is allowed, see [IAM JSON Policy Evaluation Logic \(p. 458\)](#).

When you write an IAM policy, you can control access to the following:

- [AWS for Principals \(p. 287\)](#) – Control what the person making the request (the [principal \(p. 4\)](#)) is allowed to do.
- [IAM Identities \(p. 288\)](#) – Control which IAM identities (groups, users, and roles) can be accessed and how.
- [IAM Policies \(p. 291\)](#) – Control who can create, edit, and delete customer managed policies, and who can attach and detach all managed policies.
- [AWS Resources \(p. 294\)](#) – Control who has access to resources using an identity-based policy or a resource-based policy.
- [AWS Accounts \(p. 294\)](#) – Control whether a request is allowed only for members of a specific account.

Policies let you specify who has access to AWS resources, and what actions they can perform on those resources. Every IAM user starts with no permissions. In other words, by default, users can do nothing, not even view their own access keys. To give a user permission to do something, you can add the permission to the user (that is, attach a policy to the user). Or you can add the user to a group that has the desired permission.

For example, you might grant a user permission to list his or her own access keys. You might also expand that permission and also let each user create, update, and delete their own keys.

When you give permissions to a group, all users in that group get those permissions. For example, you can give the Admins group permission to perform any of the IAM actions on any of the AWS account resources. Another example: You can give the Managers group permission to describe the AWS account's Amazon EC2 instances.

For information about how to delegate basic permissions to your users, groups, and roles, see [Permissions Required to Access IAM Resources \(p. 372\)](#). For additional examples of policies that illustrate basic permissions, see [Example Policies for Administering IAM Resources \(p. 376\)](#).

Controlling Access for Principals

You can use identity-based policies to control what the person making the request (the principal) is allowed to do. To do this, you must attach a policy to that person's identity (group, user, or role).

For example, assume that you want the user Zhang Wei to have full access to Amazon EC2, read-only access to Amazon S3, and the ability to change his own password in the AWS Management Console. You can create three different policies so that you can later break them up if you need one set of permissions for a different user. Or you can put all the permissions together in a single policy, and then attach that policy to the IAM user that is named Zhang Wei. You could also attach a policy to a group to which Zhang Wei belongs, or a role that Zhang Wei can assume. As a result, when Zhang views the contents of an S3 bucket or starts a new EC2 instance, his requests are allowed. If he tries to create a new S3 bucket or change the contents of an existing bucket, his request is denied because he doesn't have permission.

For more information, refer to these resources:

- To learn more about creating an IAM policy that you can attach to a principal, see [Creating IAM Policies \(p. 317\)](#).
- To learn how to attach an IAM policy to a principal, see [Attaching IAM Policies \(Console\) \(p. 331\)](#).
- To see an example policy for granting full access to EC2, see [Amazon EC2: Allows Full EC2 Access Within a Specific Region, Programmatically and in the Console \(p. 304\)](#).
- To allow read-only access to an S3 bucket, use the first two statements of the following example policy: [Amazon S3: Allows Read and Write Access to a Specific S3 Bucket, Programmatically and in the Console \(p. 315\)](#).
- To see an example policy for allowing users to rotate their credentials, see [IAM: Allows IAM Users to Rotate Their Own Credentials Programmatically and in the Console \(p. 309\)](#).

Controlling Access to Identities

You can use IAM policies to control what all your users can do to an identity by creating a policy that you attach to all users through the use of a group. To do this, create a policy that limits what can be done to an identity, or who can access it.

For example, you can create a group named **AllUsers**, and then attach that group to all users. When you create the group, you might give all your users access to rotate their credentials as described in the previous section. You can then create a policy that denies access to change the group unless the user name is included in the condition of the policy. But that part of the policy only denies access to anyone except those users listed. You also have to include permissions to allow all the group management actions for everyone in the group. Finally, you attach this policy to the group so that it is applied to all users. As a result, when a user not specified in the policy tries to make changes to the group, the request is denied.

To create this policy with the visual editor

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. Choose **Create policy**.
4. On the **Visual editor** tab, choose **Choose a service** to get started. Then choose **IAM**.
5. Choose **Select actions** and then type **group** in the search box. The visual editor shows all the IAM actions that contain the word **group**. Select all of the check boxes.
6. Choose **Resources** to specify resources for your policy. Based on the actions you chose, you should see **group**, **group-path**, and **user** resource types.
 - **group** – Choose **Add ARN**. For **Resource**, select the check box next to **Any**. For **Group Name With Path**, type the group name **AllUsers**. Then choose **Add**.
 - **group-path** – Select the check box next to **Any**.

- **user** – Select the check box next to **Any**.

One of the actions that you chose, `ListGroups`, does not support using specific resources. You do not have to choose **All resources** for that action. When you save your policy or view the policy on the **JSON** tab, you can see that IAM automatically creates a new permission block granting this action permission on all resources.

7. To add another permission block, choose **Add additional permissions**.
8. Choose **Choose a service** and then choose **IAM**.
9. Choose **Select actions** and then choose **Switch to deny permissions**. When you do that, the entire block is used to deny permissions.
10. Type **group** in the search box. The visual editor shows you all the IAM actions that contain the word `group`. Select the check boxes next to the following actions:
 - **CreateGroup**
 - **DeleteGroup**
 - **RemoveUserFromGroup**
 - **AttachGroupPolicy**
 - **DeleteGroupPolicy**
 - **DetachGroupPolicy**
 - **PutGroupPolicy**
 - **UpdateGroup**
11. Choose **Resources** to specify the resources for your policy. Based on the actions that you chose, you should see the **group** resource type. Choose **Add ARN**. For **Resource**, select the check box next to **Any**. For **Group Name With Path**, type the group name **AllUsers**. Then choose **Add**.
12. Choose **Specify request conditions (optional)** and then choose **Add condition**. Complete the form with the following values:
 - **Key** – Choose **aws:username**
 - **Qualifier** – Choose **For any value in request**
 - **Operator** – Choose **StringNotEquals**
 - **Value** – Type **srodriguez** and then choose **Add another condition value**. Type **mjackson** and then choose **Add another condition value**. Type **adesai** and then choose **Add**.

This condition ensures that access will be denied to the specified group management actions when the user making the call is not included in the list. Because this explicitly denies permission, it overrides the previous block that allowed those users to call the actions. Users on the list are not denied access, and they are granted permission in the first permission block, so they can fully manage the group.

13. When you are finished, choose **Review policy**.

Note

You can switch between the **Visual editor** and **JSON** tabs any time. However, if you make changes or choose **Review policy** in the **Visual editor** tab, IAM might restructure your policy to optimize it for the visual editor. For more information, see [Policy Restructuring \(p. 387\)](#).

14. On the **Review policy** page, for the **Name**, type **LimitAllUserGroupManagement**. For the **Description**, type **Allows all users Read-only access to a specific group, and allows only specific users access to make changes to the group**. Review the policy summary to make sure that you have granted the intended permissions. Then choose **Create policy** to save your new policy.
15. Attach the policy to your group. For more information, see [Attaching and Detaching IAM Policies \(p. 331\)](#).

Alternatively, you can create the same policy using this example JSON policy document. For more information, see [the section called “Create a Policy on the JSON Tab” \(p. 320\)](#).

Example Example policy that allows all users Read-only access to a specific group, and allows only specific users access to make changes to the group

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowAllUsersToListAllGroups",  
            "Effect": "Allow",  
            "Action": "iam>ListGroups",  
            "Resource": "arn:aws:iam::*:*"  
        },  
        {  
            "Sid": "AllowAllUsersToViewAndManageThisGroup",  
            "Effect": "Allow",  
            "Action": [  
                "iam>CreateGroup",  
                "iam>DeleteGroup",  
                "iam>ListGroupPolicies",  
                "iam>UpdateGroup",  
                "iam>GetGroup",  
                "iam>RemoveUserFromGroup",  
                "iam>AddUserToGroup",  
                "iam>ListGroupsForUser",  
                "iam>AttachGroupPolicy",  
                "iam>DetachGroupPolicy",  
                "iam>ListAttachedGroupPolicies",  
                "iam>GetGroupPolicy",  
                "iam>DeleteGroupPolicy",  
                "iam>PutGroupPolicy"  
            ],  
            "Resource": [  
                "arn:aws:iam::*:user/*",  
                "arn:aws:iam::*:group/AllUsers"  
            ]  
        },  
        {  
            "Sid": "LimitGroupManagementAccessToSpecificUsers",  
            "Effect": "Deny",  
            "Action": [  
                "iam>CreateGroup",  
                "iam>RemoveUserFromGroup",  
                "iam>DeleteGroup",  
                "iam>AttachGroupPolicy",  
                "iam>UpdateGroup",  
                "iam>DetachGroupPolicy",  
                "iam>DeleteGroupPolicy",  
                "iam>PutGroupPolicy"  
            ],  
            "Resource": "arn:aws:iam::*:group/AllUsers",  
            "Condition": {  
                "ForAnyValue:StringNotEquals": {  
                    "aws:username": [  
                        "srodriguez",  
                        "mjackson",  
                        "adesai"  
                    ]  
                }  
            }  
        }  
    ]  
}
```

}

Controlling Access to Policies

You can control how your users can apply AWS managed policies. To do this, attach this policy to all your users. Ideally, you can do this using a group.

For example, you might create a policy that allows users to attach only the [IAMUserChangePassword](#) and [PowerUserAccess](#) AWS managed policies to a new IAM user, group, or role.

For customer managed policies, you can control who can create, update, and delete these policies. You can control who can attach and detach policies to and from principal entities (groups, users, and roles). You can also control which policies a user can attach or detach, and to and from which entities.

For example, you can give permissions to an account administrator to create, update, and delete policies. Then you give permissions to a team leader or other limited administrator to attach and detach these policies to and from principal entities that the limited administrator manages.

For more information, refer to these resources:

- To learn more about creating an IAM policy that you can attach to a principal, see [Creating IAM Policies \(p. 317\)](#).
- To learn how to attach an IAM policy to a principal, see [Attaching IAM Policies \(Console\) \(p. 331\)](#).
- To see an example policy for limiting the use of managed policies, see [IAM: Limits Managed Policies That Can Be Applied to a New IAM User, Group, or Role \(p. 309\)](#).

Controlling Permissions for Creating, Updating, and Deleting Customer Managed Policies

You can use [IAM policies \(p. 275\)](#) to control who is allowed to create, update, and delete customer managed policies in your AWS account. The following list contains API operations that pertain directly to creating, updating, and deleting policies or policy versions:

- [CreatePolicy](#)
- [CreatePolicyVersion](#)
- [DeletePolicy](#)
- [DeletePolicyVersion](#)
- [SetDefaultPolicyVersion](#)

The API operations in the preceding list correspond to actions that you can allow or deny—that is, permissions that you can grant—using an IAM policy.

The following example shows a policy that allows a user to create, update (that is, create a new policy version), delete, and set a default version for all customer managed policies in the AWS account. The example policy also allows the user to list policies and get policies. To learn how to create a policy using this example JSON policy document, see [the section called “Create a Policy on the JSON Tab” \(p. 320\)](#).

Example policy that allows creating, updating, deleting, listing, getting, and setting the default version for all policies

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

```
    "Effect": "Allow",
    "Action": [
        "iam:CreatePolicy",
        "iam:CreatePolicyVersion",
        "iam>DeletePolicy",
        "iam>DeletePolicyVersion",
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam>ListPolicies",
        "iam>ListPolicyVersions",
        "iam:SetDefaultPolicyVersion"
    ],
    "Resource": "*"
}
```

You can create policies that limit the use of these API operations to affect only the managed policies that you specify. For example, you might want to allow a user to set the default version and delete policy versions, but only for specific customer managed policies. You do this by specifying the policy ARN in the `Resource` element of the policy that grants these permissions.

The following example shows a policy that allows a user to delete policy versions and set the default version, but only for the customer managed policies that include the path `/TEAM-A/`. The customer managed policy ARN is specified in the `Resource` element of the policy (in this example the ARN includes a path and a wildcard and thus matches all customer managed policies that include the path `/TEAM-A/`). To learn how to create a policy using this example JSON policy document, see [the section called “Create a Policy on the JSON Tab” \(p. 320\)](#).

For more information about using paths in the names of customer managed policies, see [Friendly Names and Paths \(p. 410\)](#).

Example policy that allows deleting policy versions and setting the default version for only specific policies

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam>DeletePolicyVersion",
            "iam:SetDefaultPolicyVersion"
        ],
        "Resource": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:policy/TEAM-A/*"
    }
}
```

Controlling Permissions for Attaching and Detaching Managed Policies

You can also use IAM policies to allow users to work with only specific managed policies. In effect, you can control which permissions a user is allowed to grant to other principal entities.

The following list shows API operations that pertain directly to attaching and detaching managed policies to and from principal entities:

- [AttachGroupPolicy](#)
- [AttachRolePolicy](#)
- [AttachUserPolicy](#)
- [DetachGroupPolicy](#)

- [DetachRolePolicy](#)
- [DetachUserPolicy](#)

You can create policies that limit the use of these API operations to affect only the specific managed policies and/or principal entities that you specify. For example, you might want to allow a user to attach managed policies, but only the managed policies that you specify. Or, you might want to allow a user to attach managed policies, but only to the principal entities that you specify.

The following example policy allows a user to attach managed policies to only the groups and roles that include the path /TEAM-A/. The group and role ARNs are specified in the `Resource` element of the policy. (In this example the ARNs include a path and a wildcard character and thus match all groups and roles that include the path /TEAM-A/). To learn how to create a policy using this example JSON policy document, see [the section called “Create a Policy on the JSON Tab” \(p. 320\)](#).

Example policy that allows attaching managed policies to only specific groups or roles

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:AttachGroupPolicy",
        "iam:AttachRolePolicy"
      ],
      "Resource": [
        "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:group/TEAM-A/*",
        "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:role/TEAM-A/*"
      ]
    }
  ]
}
```

You can further limit the actions in the preceding example to affect only specific policies—that is, you can control which permissions a user is allowed to attach to other principal entities—by adding a condition to the policy.

In the following example, the condition ensures that the `AttachGroupPolicy` and `AttachRolePolicy` permissions are allowed only when the policy being attached matches one of the specified policies. The condition uses the `iam:PolicyArn` condition key (p. 438) to determine which policy or policies are allowed to be attached. The following example policy expands on the previous example by allowing a user to attach only the managed policies that include the path /TEAM-A/ to only the groups and roles that include the path /TEAM-A/. To learn how to create a policy using this example JSON policy document, see [the section called “Create a Policy on the JSON Tab” \(p. 320\)](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:AttachGroupPolicy",
        "iam:AttachRolePolicy"
      ],
      "Resource": [
        "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:group/TEAM-A/*",
        "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:role/TEAM-A/*"
      ],
      "Condition": {"ArnLike":
        {"iam:PolicyArn": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:policy/TEAM-A/*"}
      }
    }
  ]
}
```

}

This policy uses the `ArnLike` condition operator because the ARN includes a wildcard character. For a specific ARN, use the `ArnEquals` condition operator. For more information about `ArnLike` and `ArnEquals`, see [Amazon Resource Name \(ARN\) Condition Operators \(p. 445\)](#) in the *Condition Types* section of the *Policy Element Reference*.

For example, you can limit the use of actions to involve only the managed policies that you specify. You do this by specifying the policy ARN in the `Condition` element of the policy that grants these permissions. For example, to specify the ARN of a customer managed policy:

```
"Condition": {"ArnEquals":  
    {"iam:PolicyArn": "arn:aws:iam::123456789012:policy/POLICY-NAME"}  
}
```

You can also specify the ARN of an AWS managed policy in a policy's `Condition` element. The ARN of an AWS managed policy uses the special alias `aws` in the policy ARN instead of an account ID, as in this example:

```
"Condition": {"ArnEquals":  
    {"iam:PolicyArn": "arn:aws:iam::aws:policy/AmazonEC2FullAccess"}  
}
```

Controlling Access to Resources

You can control access to resources using an identity-based policy or a resource-based policy. In an identity-based policy, you attach the policy to an identity and specify what resources that identity can access. In a resource-based policy, you attach a policy to the resource that you want to control. In the policy, you specify which principals can access that resource. For more information about both types of policies, see [Identity-Based Policies and Resource-Based Policies \(p. 285\)](#).

For more information, refer to these resources:

- To learn more about creating an IAM policy that you can attach to a principal, see [Creating IAM Policies \(p. 317\)](#).
- To learn how to attach an IAM policy to a principal, see [Attaching IAM Policies \(Console\) \(p. 331\)](#).
- Amazon S3 supports using resource-based policies on their buckets. For more information, see [Bucket Policy Examples](#).

Resource Creators Do Not Automatically Have Permissions

If you sign in using the AWS account root user credentials, you have permission to perform any action on resources that belong to the account. However, this isn't true for IAM users. An IAM user might be granted access to create a resource, but the user's permissions, even for that resource, are limited to what's been explicitly granted. This means that just because you create a resource, such as an IAM role, you do not automatically have permission to edit or delete that role. Additionally, your permission can be revoked at any time by the account owner or by another user who has been granted access to manage your permissions.

Controlling Access to Principals in a Specific Account

You can directly grant IAM users in your own account access to your resources. If users from another account need access to your resources, you can create an IAM role, which is an entity that includes permissions but that isn't associated with a specific user. Users from other accounts can then assume the

role and access resources according to the permissions you've assigned to the role. For more information, see [Providing Access to an IAM User in Another AWS Account That You Own \(p. 138\)](#).

Note

For services that support resource-based policies as described in [Identity-Based Policies and Resource-Based Policies \(p. 285\)](#) (such as Amazon S3, Amazon SNS, and Amazon SQS), an alternative to using roles is to attach a policy to the resource (bucket, topic, or queue) that you want to share. The resource-based policy can specify the AWS account that has permissions to access the resource.

Example Policies

A [policy \(p. 275\)](#) is an entity in AWS that, when attached to an identity or resource, defines their permissions. AWS evaluates these policies when a principal, such as a user, makes a request. Permissions in the policies determine whether the request is allowed or denied. Policies are stored in AWS as JSON documents that are attached to principals as *identity-based policies* or to resources as *resource-based policies*. You can attach an identity-based policy to a principal (or identity), such as an IAM group, user, or role. Identity-based policies include AWS managed policies, customer managed policies, and inline policies. To learn how to create an IAM policy using these example JSON policy documents, see [the section called "Create a Policy on the JSON Tab" \(p. 320\)](#).

By default all requests are denied, so you must provide access to the services, actions, and resources that you intend for the identity to access. If you also want to allow access to complete the specified actions in the IAM console, you need to provide additional permissions.

The following library of policies can help you define permissions for your IAM identities. After you find the policy that you need, choose **View this policy** to view the JSON for the policy. You can use the JSON policy document as a template for your own policies.

Note

If you would like to submit a policy to be included in this reference guide, use the **Feedback** button at the bottom of this page.

Example Policies: AWS

- Allows access during a specific range of dates ([View this policy \(p. 297\)](#))
- Allows specific access when using MFA during a specific range of dates ([View this policy \(p. 297\)](#))
- Denies access to AWS based on the source IP address ([View this policy \(p. 297\)](#))

Example Policies: AWS CodeCommit

- Allows Read access to an AWS CodeCommit repository, programmatically and in the console ([View this policy \(p. 298\)](#))

Example Policies: AWS Data Pipeline

- Denies access to pipelines that a user did not create ([View this policy \(p. 298\)](#))

Example Policies: Amazon DynamoDB

- Allows access to a specific Amazon DynamoDB table ([View this policy \(p. 299\)](#))
- Allows access to specific Amazon DynamoDB columns ([View this policy \(p. 300\)](#))
- Allows row-level access to Amazon DynamoDB based on an Amazon Cognito ID ([View this policy \(p. 300\)](#))

Example Policies: Amazon EC2

- Allows an Amazon EC2 instance to attach or detach volumes ([View this policy \(p. 301\)](#))
- Allows attaching or detaching Amazon EBS volumes to Amazon EC2 instances based on tags ([View this policy \(p. 302\)](#))
- Allows launching Amazon EC2 instances in a specific subnet, programmatically and in the console ([View this policy \(p. 302\)](#))
- Allows managing Amazon EC2 security groups associated with a specific VPC, programmatically and in the console ([View this policy \(p. 303\)](#))
- Allows starting or stopping Amazon EC2 instances a user has tagged, programmatically and in the console ([View this policy \(p. 303\)](#))
- Allows full Amazon EC2 access within a specific region, programmatically and in the console ([View this policy \(p. 304\)](#))
- Allows starting or stopping a specific Amazon EC2 instance and modifying a specific security group, programmatically and in the console ([View this policy \(p. 304\)](#))
- Limits terminating Amazon EC2 instances to a specific IP address range ([View this policy \(p. 305\)](#))

Example Policies: AWS Identity and Access Management (IAM)

- Allows access to the policy simulator API ([View this policy \(p. 306\)](#))
- Allows access to the policy simulator console ([View this policy \(p. 306\)](#))
- Allows using the policy simulator API for users with a specific path ([View this policy \(p. 307\)](#))
- Allows using the policy simulator console for users with a specific path ([View this policy \(p. 307\)](#))
- Allows IAM users to self-manage an MFA device ([View this policy \(p. 308\)](#))
- Allows IAM users to rotate their own credentials, programmatically and in the console ([View this policy \(p. 309\)](#))
- Limits managed policies that can be applied to a new IAM user, group, or role ([View this policy \(p. 309\)](#))

Example Policies: Amazon RDS

- Allows full Amazon RDS database access within a specific region ([View this policy \(p. 310\)](#))
- Allows restoring Amazon RDS databases, programmatically and in the console ([View this policy \(p. 310\)](#))
- Allows tag owners full access to Amazon RDS resources that they have tagged ([View this policy \(p. 311\)](#))

Example Policies: Amazon S3

- Allows an Amazon Cognito user to access objects in their own Amazon S3 bucket ([View this policy \(p. 313\)](#))
- Allows IAM users to access their own home directory in Amazon S3, programmatically and in the console ([View this policy \(p. 314\)](#))
- Allows a user to manage a single Amazon S3 bucket and denies every other AWS action and resource ([View this policy \(p. 314\)](#))
- Allows Read and Write access to a specific Amazon S3 bucket ([View this policy \(p. 315\)](#))
- Allows Read and Write access to a specific Amazon S3 bucket, programmatically and in the console ([View this policy \(p. 315\)](#))

AWS: Allows Access Within Specific Dates

This example shows how you might create a policy that allows access to the ACTION-NAME action in the service named SERVICE-NAME. Access is restricted to actions that occur between July 1, 2017 and December 31, 2017 (UTC), inclusive. To use this policy, replace the red text in the example policy with your own information.

To learn about using multiple conditions within the Condition block of an IAM policy, see [Multiple Values in a Condition \(p. 439\)](#).

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Action": "<SERVICE-NAME>:<ACTION-NAME>",  
        "Resource": "*",  
        "Condition": {  
            "DateGreaterThan": {"aws:CurrentTime": "2017-07-01T00:00:00Z"},  
            "DateLessThan": {"aws:CurrentTime": "2017-12-31T23:59:59Z"}  
        }  
    }  
}
```

AWS: Allows Specific Access Using MFA Within Specific Dates

This example shows how you might create a policy that uses multiple conditions, which are evaluated using a logical AND. It allows full access to the service named SERVICE-NAME-1, and access to the ACTION-NAME-A and ACTION-NAME-B actions in the service named SERVICE-NAME-2. These actions are allowed only when the user is authenticated using [multifactor authentication \(MFA\)](#). Access is restricted to actions that occur between July 1, 2017 and December 31, 2017 (UTC), inclusive. To use this policy, replace the red text in the example policy with your own information.

To learn about using multiple conditions within the Condition block of an IAM policy, see [Multiple Values in a Condition \(p. 439\)](#).

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Action": [  
            "<SERVICE-NAME-1>:*",  
            "<SERVICE-NAME-2>:<ACTION-NAME-A>",  
            "<SERVICE-NAME-2>:<ACTION-NAME-B>"  
        ],  
        "Resource": "*",  
        "Condition": {  
            "Bool": {"aws:MultiFactorAuthPresent": true},  
            "DateGreaterThan": {"aws:CurrentTime": "2017-07-01T00:00:00Z"},  
            "DateLessThan": {"aws:CurrentTime": "2017-12-31T23:59:59Z"}  
        }  
    }  
}
```

AWS: Denies Access to AWS Based on the Source IP

This example shows how you might create a policy that denies access to all AWS actions in the account when the request comes from outside the specified IP range. The policy is useful when the IP addresses for your company are within the specified ranges. This policy also provides the permissions necessary to complete this action on the console. To use this policy, replace the red text in the example policy with your own information.

The `aws:SourceIp` condition key denies access to an AWS service, such as AWS CloudFormation, that makes calls on your behalf. For more information about using the `aws:SourceIp` condition key, see [Available Global Condition Keys \(p. 477\)](#).

Important

This policy does not allow any actions. Use this policy in combination with other policies that allow specific actions.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Action": "*",  
            "Resource": "*",  
            "Condition": {"NotIpAddress": {"aws:SourceIp": [  
                "192.0.2.0/24",  
                "203.0.113.0/24"  
            ]}}  
        }  
    ]  
}
```

AWS CodeCommit: Allows Read Access to an AWS CodeCommit Repository, Programmatically and in the Console

This example shows how you might create a policy that allows Read access to a specific CodeCommit repository. This policy also provides the permissions necessary to complete this action on the console. To use this policy, replace the red text in the example policy with your own information.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "codecommit:GitPull"  
            ],  
            "Resource": "arn:aws:codecommit:<REGION>:<ACCOUNTNUMBER>:<REPO-NAME>"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "codecommit:Get*",  
                "codecommit:BatchGetRepositories",  
                "codecommit>List*"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

AWS Data Pipeline: Denies Access to DataPipeline Pipelines That a User Did Not Create

This example shows how you might create a policy that denies access to pipelines that a user did not create. If the value of the `PipelineCreator` field matches the IAM user name, then the specified actions are not denied.

Important

This policy does not allow any actions. Use this policy in combination with other policies that allow specific actions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExplicitDenyIfNotTheOwner",
            "Effect": "Deny",
            "Action": [
                "datapipeline:ActivatePipeline",
                "datapipeline:AddTags",
                "datapipeline:DeactivatePipeline",
                "datapipeline>DeletePipeline",
                "datapipeline:DescribeObjects",
                "datapipeline:EvaluateExpression",
                "datapipeline:GetPipelineDefinition",
                "datapipeline:PollForTask",
                "datapipeline:PutPipelineDefinition",
                "datapipeline:QueryObjects",
                "datapipeline:RemoveTags",
                "datapipeline:ReportTaskProgress",
                "datapipeline:ReportTaskRunnerHeartbeat",
                "datapipeline:SetStatus",
                "datapipeline:SetTaskStatus",
                "datapipeline:ValidatePipelineDefinition"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "datapipeline:PipelineCreator": "${aws:userid}"
                }
            }
        }
    ]
}
```

Amazon DynamoDB: Allows Access to a Specific Table

This example shows how you might create a policy that allows full access to a DynamoDB table with the specified name. To use this policy, replace the red text in the example policy with your own information.

Important

This policy allows all actions that can be performed on a DynamoDB table. To review these actions, see [DynamoDB API Permissions: Actions, Resources, and Conditions Reference](#) in the [Amazon DynamoDB Developer Guide](#). You could provide the same permissions by listing each individual action. However, if you use the "Action": "dynamodb:*" element, then you will not need to update your policy if DynamoDB adds a new action for DynamoDB tables.

This policy allows actions only on DynamoDB tables that exist with the specified name. To allow your users Read access to everything in DynamoDB, you can also attach the [AmazonDynamoDBReadOnlyAccess](#) AWS managed policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:<REGION>:<ACCOUNTNUMBER>:table/<TABLE-NAME>"
        }
    ]
}
```

Amazon DynamoDB: Allows Access to Specific Columns

This example shows how you might create a policy that allows access to the specific DynamoDB columns. To use this policy, replace the red text in the example policy with your own information.

The dynamodb:Select requirement prevents the API action from returning any attributes that aren't allowed, such as from an index projection. To learn more about DynamoDB condition keys, see [Specifying Conditions: Using Condition Keys](#) in the *Amazon DynamoDB Developer Guide*. To learn about using multiple conditions or multiple condition keys within the Condition block of an IAM policy, see [Multiple Values in a Condition \(p. 439\)](#).

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:GetItem",  
                "dynamodb:BatchGetItem",  
                "dynamodb:Query",  
                "dynamodb:PutItem",  
                "dynamodb:UpdateItem",  
                "dynamodb:DeleteItem",  
                "dynamodb:BatchWriteItem"  
            ],  
            "Resource": [  
                "arn:aws:dynamodb:<REGION>:<ACCOUNTNUMBER>:table/<TABLE-NAME>"  
            ],  
            "  
            "Condition": {  
                "ForAllValues:StringEquals": {  
                    "dynamodb:Attributes": [  
                        "<COLUMN-NAME-1>",  
                        "<COLUMN-NAME-2>",  
                        "<COLUMN-NAME-3>"  
                    ]  
                },  
                "StringEqualsIfExists": {  
                    "dynamodb:Select": "SPECIFIC_ATTRIBUTES"  
                }  
            }  
        }  
    ]  
}
```

Amazon DynamoDB: Allows Row-Level Access to DynamoDB Based on an Amazon Cognito ID

This example shows how you might create a policy that allows row-level access to a specific DynamoDB table based on an Amazon Cognito ID. To use this policy, replace the red text in the example policy with your own information.

To use this policy, you must structure your DynamoDB table so the Cognito user ID is the partition key. For more information, see [Creating a Table](#) in the *Amazon DynamoDB Developer Guide*.

To learn more about DynamoDB condition keys, see [Specifying Conditions: Using Condition Keys](#) in the *Amazon DynamoDB Developer Guide*.

```
{
```

```

"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "dynamodb>DeleteItem",
            "dynamodb>GetItem",
            "dynamodb>PutItem",
            "dynamodb>Query",
            "dynamodb>UpdateItem"
        ],
        "Resource": [
            "arn:aws:dynamodb:<REGION>:<ACCOUNTNUMBER>:table/<TABLE-NAME>"
        ],
        "Condition": {
            "ForAllValues:StringEquals": {
                "dynamodb:LeadingKeys": [
                    "${cognito-identity.amazonaws.com:sub}"
                ]
            }
        }
    }
]
}

```

Amazon EC2: Allows an EC2 Instance to Attach or Detach Volumes

This example shows how you might create a policy that can be attached to a service role. The policy allows the specified EC2 instance to attach or detach volumes. The instance is specified with an ARN in the `Condition` element. To use this policy, replace the red text in the example policy with your own information.

Amazon EC2 instances can run AWS commands with permissions granted by an [AWS service role for an EC2 instance \(p. 136\)](#) that is attached to the instance profile. You can attach this policy to the role, or add this statement to an existing policy. Only the instance identified by `INSTANCE-ID` can attach or detach volumes to instances in the account, including its own. Other statement elements that might exist in a larger policy are not impacted by the restriction of this one statement. For more information about creating IAM policies to control access to Amazon EC2 resources, see [Controlling Access to Amazon EC2 Resources](#) in the *Amazon EC2 User Guide for Linux Instances*.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachVolume",
                "ec2:DetachVolume"
            ],
            "Resource": [
                "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:volume/*",
                "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:instance/*"
            ],
            "Condition": {
                "ArnEquals": {
                    "ec2:SourceInstanceARN": "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:instance/<INSTANCE-ID>"
                }
            }
        }
    ]
}

```

```
}
```

Amazon EC2: Attach or Detach Amazon EBS Volumes to EC2 Instances Based on Tags

This example shows how you might create a policy that allows EBS volume owners to attach or detach their EBS volumes defined using the tag `volume_user` to EC2 instances that are tagged as development instances (`department=dev`). To use this policy, replace the red text in the example policy with your own information.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachVolume",
                "ec2:DetachVolume"
            ],
            "Resource": "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:instance/*",
            "Condition": {"StringEquals": {"ec2:ResourceTag/department": "dev"}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachVolume",
                "ec2:DetachVolume"
            ],
            "Resource": "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:volume/*",
            "Condition": {"StringEquals": {"ec2:ResourceTag/volume_user": "${aws:username}"}}
        }
    ]
}
```

Amazon EC2: Allows Launching EC2 Instances in a Specific Subnet, Programmatically and in the Console

This example shows how you might create a policy that allows listing information for all EC2 objects and launching EC2 instances in a specific subnet. This policy also provides the permissions necessary to complete this action on the console. To use this policy, replace the red text in the example policy with your own information.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
                "ec2:GetConsole*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:subnet/subnet-<SUBNET-ID>",
                "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:network-interface/*",

```

```

        "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:instance/*",
        "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:volume/*",
        "arn:aws:ec2:<REGION>::image/ami-*",
        "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:key-pair/*",
        "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:security-group/*"
    ]
}
]
}

```

Amazon EC2: Allows Managing EC2 Security Groups Associated With a Specific VPC, Programmatically and in the Console

This example shows how you might create a policy that allows managing Amazon EC2 security groups associated with a specific virtual private cloud (VPC). This policy also provides the permissions necessary to complete this action on the console. To use this policy, replace the red text in the example policy with your own information.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:DeleteSecurityGroup",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:security-group/*",
            "Condition": {
                "StringEquals": {
                    "ec2:Vpc": "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:vpc/vpc-<VPC-ID>"
                }
            }
        },
        {
            "Action": [
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSecurityGroupReferences",
                "ec2:DescribeStaleSecurityGroups",
                "ec2:DescribeVpcs"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}

```

Amazon EC2: Allows Starting or Stopping EC2 Instances a User Has Tagged, Programmatically and in the Console

This example shows how you might create a policy that allows an IAM user to start or stop EC2 instances, but only if the instance tag `Owner` has the value of that user's user name. This policy also provides the permissions necessary to complete this action on the console. To use this policy, replace the red text in the example policy with your own information.

```
{
}
```

```

    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:instance/*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/Owner": "${aws:username}"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DescribeInstances",
            "Resource": "*"
        }
    ]
}

```

Amazon EC2: Allows Full EC2 Access Within a Specific Region, Programmatically and in the Console

This example shows how you might create a policy that allows full EC2 access within a specific region. This policy also provides the permissions necessary to complete this action on the console. To use this policy, replace the red text in the example policy with your own information.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "ec2:*",
            "Resource": "*",
            "Effect": "Allow",
            "Condition": {
                "StringEquals": {
                    "ec2:Region": "<REGION>"
                }
            }
        }
    ]
}

```

Amazon EC2: Allows Starting or Stopping an EC2 Instance and Modifying a Security Group, Programmatically and in the Console

This example shows how you might create a policy that allows starting or stopping a specific EC2 instance and modifying a specific security group. This policy also provides the permissions necessary to complete this action on the console. To use this policy, replace the red text in the example policy with your own information.

```

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

```

```
{
    "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSecurityGroupReferences",
        "ec2:DescribeStaleSecurityGroups"
    ],
    "Resource": "*",
    "Effect": "Allow"
},
{
    "Action": [
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:StartInstances",
        "ec2:StopInstances"
    ],
    "Resource": [
        "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:instance/i-<INSTANCE-ID>",
        "arn:aws:ec2:<REGION>:<ACCOUNTNUMBER>:security-group/sg-<SECURITY-GROUP-ID>"
    ],
    "Effect": "Allow"
}
]
```

Amazon EC2: Limits Terminating EC2 Instances to an IP Address Range

This example shows how you might create a policy that limits EC2 instances by allowing the action, but explicitly denying access when the request comes from outside the specified IP range. The policy is useful when the IP addresses for your company are within the specified ranges. To use this policy, replace the red text in the example policy with your own information.

If this policy is used in combination with other policies that allow the `ec2:TerminateInstances` action (such as the [AmazonEC2FullAccess](#) AWS managed policy), then access is denied. This is because an explicit deny statement takes precedence over allow statements. For more information, see [the section called “Determining Whether a Request is Allowed or Denied” \(p. 459\)](#).

Important

The `aws:SourceIp` condition key denies access to an AWS service, such as AWS CloudFormation, that makes calls on your behalf. For more information about using the `aws:SourceIp` condition key, see [Available Global Condition Keys \(p. 477\)](#).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:TerminateInstances"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [

```

```
        "ec2:TerminateInstances"
    ],
    "Condition": {"NotIpAddress": {"aws:SourceIp": [
        "192.0.2.0/24",
        "203.0.113.0/24"
    ]}},
    "Resource": [
        "*"
    ]
}
]
```

IAM: Access the Policy Simulator API

This example shows how you might create a policy that allows using the policy simulator API for policies attached to a user, group, or role in the current AWS account. This policy also allows access to simulate less sensitive policies passed to the API as strings.

```
{
    "Version" : "2012-10-17",
    "Statement" : [
        {
            "Action" : [
                "iam:GetContextKeysForCustomPolicy",
                "iam:GetContextKeysForPrincipalPolicy",
                "iam:SimulateCustomPolicy",
                "iam:SimulatePrincipalPolicy"
            ],
            "Effect" : "Allow",
            "Resource" : "*"
        }
    ]
}
```

Note

To allow a user to access the policy simulator console to simulate policies attached to a user, group, or role in the current AWS account, see [IAM: Access the Policy Simulator Console \(p. 306\)](#).

IAM: Access the Policy Simulator Console

This example shows how you might create a policy that allows using the policy simulator console for policies attached to a user, group, or role in the current AWS account.

You can access the IAM Policy Simulator console at: <https://policysim.aws.amazon.com/>

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:GetGroup",
                "iam:GetGroupPolicy",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:GetUser",
                "iam:GetUserPolicy",
                "iam>ListAttachedGroupPolicies",
                "iam:GetServerCertificate"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

```
        "iam>ListAttachedRolePolicies",
        "iam>ListAttachedUserPolicies",
        "iam>ListGroups",
        "iam>ListGroupPolicies",
        "iam>ListGroupsForUser",
        "iam>ListRolePolicies",
        "iam>ListRoles",
        "iam>ListUserPolicies",
        "iam>ListUsers"
    ],
    "Effect": "Allow",
    "Resource": "*"
}
]
}
```

IAM: Access the Policy Simulator API Based on User Path

This example shows how you might create a policy that allows using the policy simulator API only for those users that have the path `USER-PATH-NAME`. To use this policy, replace the red text in the example policy with your own information.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:GetContextKeysForPrincipalPolicy",
                "iam:SimulatePrincipalPolicy"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::<ACCOUNTNUMBER>:user/<USER-PATH-NAME>/*"
        }
    ]
}
```

Note

To create a policy that allows using the policy simulator console for those users that have the path `USER-PATH-NAME`, see [IAM: Access the Policy Simulator Console Based on User Path \(p. 307\)](#).

IAM: Access the Policy Simulator Console Based on User Path

This example shows how you might create a policy that allows using the policy simulator console only for those users that have the path `USER-PATH-NAME`. To use this policy, replace the red text in the example policy with your own information.

You can access the IAM Policy Simulator at: <https://pollicysim.aws.amazon.com/>

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:GetPolicy",
                "iam:GetUserPolicy"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
    ]
}
```

```
{  
    "Action": [  
        "iam:GetUser",  
        "iam>ListAttachedUserPolicies",  
        "iam>ListGroupsForUser",  
        "iam>ListUserPolicies",  
        "iam>ListUsers"  
    ],  
    "Effect": "Allow",  
    "Resource": "arn:aws:iam::<ACCOUNTNUMBER>:user/<USER-PATH-NAME>/*"  
}  
}  
}
```

IAM: Allows IAM Users to Self-Manage an MFA Device

This example shows how you might create a policy that allows IAM users to self-manage an MFA device.

Note

If you add these permissions for a user that is signed in to AWS, they might need to sign out and back in to see these changes.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iam>CreateVirtualMFADevice",  
                "iam:EnableMFADevice",  
                "iam:ResyncMFADevice",  
                "iam>DeleteVirtualMFADevice"  
            ],  
            "Resource": [  
                "arn:aws:iam::*:mfa/${aws:username}",  
                "arn:aws:iam::*:user/${aws:username}"  
            ]  
        },  
        {  
            "Sid": "AllowUsersToDeactivateTheirOwnVirtualMFADevice",  
            "Effect": "Allow",  
            "Action": [  
                "iam:DeactivateMFADevice"  
            ],  
            "Resource": [  
                "arn:aws:iam::*:mfa/${aws:username}",  
                "arn:aws:iam::*:user/${aws:username}"  
            ],  
            "Condition": {  
                "Bool": {  
                    "aws:MultiFactorAuthPresent": "true"  
                }  
            }  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iam>ListMFADevices",  
                "iam>ListVirtualMFADevices",  
                "iam>ListUsers"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

```
}
```

IAM: Allows IAM Users to Rotate Their Own Credentials Programmatically and in the Console

This example shows how you might create a policy that allows IAM users to rotate their own access keys, signing certificates, service specific credentials, and passwords. This policy also provides the permissions necessary to complete this action on the console.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam>ListUsers",
                "iam>GetAccountPasswordPolicy"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam>*AccessKey*",
                "iam>ChangePassword",
                "iam GetUser",
                "iam>*ServiceSpecificCredential*",
                "iam>*SigningCertificate"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        }
    ]
}
```

To learn how a user can change their own password in the console, see [the section called “How IAM Users Change Their Own Password” \(p. 83\)](#).

IAM: Limits Managed Policies That Can Be Applied to a New IAM User, Group, or Role

This example shows how you might create a policy that limits customer managed and AWS managed policies that can be applied to a new IAM user, group, or role. To use this policy, replace the red text in the example policy with your own information.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam>ChangePassword",
                "iam>CreateAccessKey",
                "iam>CreateLoginProfile",
                "iam>CreateUser",
                "iam>DeleteAccessKey",
                "iam>DeleteLoginProfile",
                "iam>DeleteUser",
                "iam>UpdateAccessKey",
                "iam>ListAttachedUserPolicies",
                "iam>ListPolicies",
                "iam>GetServiceLastAccess"
            ],
            "Resource": [
                "arn:aws:iam::${aws:username}:user/${aws:username}",
                "arn:aws:iam::${aws:username}:group/${aws:username}-group"
            ]
        }
    ]
}
```

```
        "iam>ListUserPolicies",
        "iam>ListGroups",
        "iam>ListGroupsForUser",
        "iam>GetPolicy",
        "iam>GetAccountSummary"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "iam>AttachUserPolicy",
        "iam>DetachUserPolicy"
    ],
    "Resource": "*",
    "Condition": {
        "ArnEquals": {
            "iam:PolicyArn": [
                "arn:aws:iam:<ACCOUNTNUMBER>:policy/<MANAGED-POLICY-NAME>",
                "arn:aws:iam:aws:policy/<AWS-MANAGED-POLICY-NAME>"
            ]
        }
    }
}
]
```

Amazon RDS: Allows Full RDS Database Access Within a Specific Region

This example shows how you might create a policy that allows full RDS database access within a specific region. To use this policy, replace the red text in the example policy with your own information.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds:*",
            "Resource": [
                "arn:aws:rds:<REGION>:<ACCOUNTNUMBER>:/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "rds:Describe*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Amazon RDS: Allows Restoring RDS Databases, Programmatically and In the Console

This example shows how you might create a policy that allows restoring RDS databases. This policy also provides the permissions necessary to complete this action on the console.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "ec2:Describe*",  
                "rds>CreateDBParameterGroup",  
                "rds>CreateDBSnapshot",  
                "rds>DeleteDBSnapshot",  
                "rds:Describe*",  
                "rds:DownloadDBLogFilePortion",  
                "rds>List*",  
                "rds:ModifyDBInstance",  
                "rds:ModifyDBParameterGroup",  
                "rds:ModifyOptionGroup",  
                "rds:RebootDBInstance",  
                "rds:RestoreDBInstanceFromDBSnapshot",  
                "rds:RestoreDBInstanceToPointInTime"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

Amazon RDS: Allows Tag Owners Full Access to RDS Resources That They Have Tagged

This example shows how you might create a policy that allows tag owners full access to RDS resources that they have tagged.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "rds:Describe*",  
                "rds>List*"  
            ],  
            "Effect": "Allow",  
            "Resource": "*"  
        },  
        {  
            "Action": [  
                "rds>DeleteDBInstance",  
                "rds:RebootDBInstance",  
                "rds:ModifyDBInstance"  
            ],  
            "Effect": "Allow",  
            "Resource": "*",  
            "Condition": {  
                "StringEqualsIgnoreCase": {  
                    "rds:db-tag/Owner": "${aws:username}"  
                }  
            }  
        },  
        {  
            "Action": [  
                "rds:ModifyOptionGroup",  
                "rds>DeleteOptionGroup"  
            ],  
            "Effect": "Allow",  
            "Resource": "*"  
        }  
    ]  
}
```

```
"Resource": "*",
"Condition": {
    "StringEqualsIgnoreCase": {
        "rds:og-tag/Owner": "${aws:username}"
    }
},
{
    "Action": [
        "rds:ModifyDBParameterGroup",
        "rds:ResetDBParameterGroup"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
        "StringEqualsIgnoreCase": {
            "rds:pg-tag/Owner": "${aws:username}"
        }
    }
},
{
    "Action": [
        "rds:AuthorizeDBSecurityGroupIngress",
        "rds:RevokeDBSecurityGroupIngress",
        "rds:DeleteDBSecurityGroup"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
        "StringEqualsIgnoreCase": {
            "rds:secgrp-tag/Owner": "${aws:username}"
        }
    }
},
{
    "Action": [
        "rds:DeleteDBSnapshot",
        "rds:RestoreDBInstanceFromDBSnapshot"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
        "StringEqualsIgnoreCase": {
            "rds:snapshot-tag/Owner": "${aws:username}"
        }
    }
},
{
    "Action": [
        "rds:ModifyDBSubnetGroup",
        "rds:DeleteDBSubnetGroup"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
        "StringEqualsIgnoreCase": {
            "rds:subgrp-tag/Owner": "${aws:username}"
        }
    }
},
{
    "Action": [
        "rds:ModifyEventSubscription",
        "rds:AddSourceIdentifierToSubscription",
        "rds:RemoveSourceIdentifierFromSubscription",
        "rds:DeleteEventSubscription"
    ]
}
```

```
        ],
        "Effect": "Allow",
        "Resource": "*",
        "Condition": {
            "StringEqualsIgnoreCase": {
                "rds:es-tag/Owner": "${aws:username}"
            }
        }
    ]
}
```

Amazon S3: Allows Amazon Cognito Users to Access Objects in Their Bucket

This example shows how you might create a policy that allows Amazon Cognito users to access objects in a specific S3 bucket. This policy allows access only to objects with a name that includes cognito, the name of the application, and the federated user's ID. To use this policy, replace the red text in the example policy with your own information.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["s3>ListBucket"],
            "Resource": ["arn:aws:s3:::<BUCKET-NAME>"],
            "Condition": {"StringLike": {"s3:prefix": ["cognito/<APPLICATION-NAME>/"]}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3>DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::<BUCKET-NAME>/cognito/<APPLICATION-NAME>/${cognito-
identity.amazonaws.com:sub}",
                "arn:aws:s3:::<BUCKET-NAME>/cognito/<APPLICATION-NAME>/${cognito-
identity.amazonaws.com:sub}/*"
            ]
        }
    ]
}
```

Amazon Cognito is an easy way to use web identity federation in your mobile app. Using Amazon Cognito, you can provide access to AWS resources for users who have signed in to your app using a third-party identity provider like Login with Amazon, Facebook, Google, or any Open-ID Connect (OIDC) compatible identity provider instead of using an IAM user. To use Amazon Cognito for web identity federation, you create a role that determines what permissions the federated user will have. You can create one role for authenticated users. If your app allows unauthenticated (guest) users, you can create a second role that defines the permissions for those users.

For more information about Amazon Cognito, see the following:

- [Amazon Cognito Identity](#) in the *AWS Mobile SDK for Android Developer Guide*
- [Amazon Cognito Identity](#) in the *AWS Mobile SDK for iOS Developer Guide*

Amazon S3: Allows IAM Users Access to Their S3 Home Directory, Programmatically and In the Console

This example shows how you might create a policy that allows IAM users to access their own home directory in S3. The home directory is a bucket that includes a home folder and folders for individual users. This policy also provides the permissions necessary to complete this action on the console. To use this policy, replace the red text in the example policy with your own information.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3>ListAllMyBuckets",  
                "s3:GetBucketLocation"  
            ],  
            "Resource": "arn:aws:s3:::*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": "s3>ListBucket",  
            "Resource": "arn:aws:s3:::<BUCKET-NAME>",  
            "Condition": {"StringLike": {"s3:prefix": [  
                "",  
                "home/",  
                "home/${aws:username}/*"  
            ]}}  
        },  
        {  
            "Effect": "Allow",  
            "Action": "s3:*",  
            "Resource": [  
                "arn:aws:s3:::<BUCKET-NAME>/home/${aws:username}",  
                "arn:aws:s3:::<BUCKET-NAME>/home/${aws:username}/*"  
            ]  
        }  
    ]  
}
```

Amazon S3: Limits Managing to a Specific S3 Bucket

This example shows how you might create a policy that limits managing an S3 bucket by allowing all S3 actions on the specific bucket, but explicitly denying access to every AWS service except Amazon S3. This policy also denies access to actions that can't be performed on an S3 bucket, such as `s3>ListAllMyBuckets` or `s3GetObject`. To use this policy, replace the red text in the example policy with your own information.

If this policy is used in combination with other policies (such as the [AmazonS3FullAccess](#) or [AmazonEC2FullAccess](#) AWS managed policies) that allow actions denied by this policy, then access is denied. This is because an explicit deny statement takes precedence over allow statements. For more information, see [the section called "Determining Whether a Request is Allowed or Denied" \(p. 459\)](#).

Warning

[NotAction](#) (p. 435) and [NotResource](#) (p. 437) are advanced policy elements that must be used with care. This policy denies access to every AWS service except Amazon S3. If you attach this policy to a user, any other policies that grant permissions to other services are ignored and access is denied.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::<BUCKET-NAME>",
      "arn:aws:s3:::<BUCKET-NAME>/*"
    ]
  },
  {
    "Effect": "Deny",
    "NotAction": "s3:*",
    "NotResource": [
      "arn:aws:s3:::<BUCKET-NAME>",
      "arn:aws:s3:::<BUCKET-NAME>/*"
    ]
  }
]
```

Amazon S3: Allows Read and Write Access to a Specific S3 Bucket

This example shows how you might create a policy that allows `Read` and `Write` access to a specific S3 bucket. To use this policy, replace the red text in the example policy with your own information.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3>ListBucket"],
      "Resource": ["arn:aws:s3:::<BUCKET-NAME>"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3>PutObject",
        "s3.GetObject"
      ],
      "Resource": ["arn:aws:s3:::<BUCKET-NAME>/*"]
    }
  ]
}
```

Note

To allow `Read` and `Write` access to a specific Amazon S3 bucket and also include additional permissions for console access, see [Amazon S3: Allows Read and Write Access to a Specific S3 Bucket, Programmatically and in the Console \(p. 315\)](#).

Amazon S3: Allows Read and Write Access to a Specific S3 Bucket, Programmatically and in the Console

This example shows how you might create a policy that allows `Read` and `Write` access to a specific S3 bucket. This policy also provides the permissions necessary to complete this action on the console. To use this policy, replace the red text in the example policy with your own information.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetBucketLocation",
            "s3>ListAllMyBuckets"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": ["s3>ListBucket"],
        "Resource": ["arn:aws:s3:::<BUCKET-NAME>"]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject"
        ],
        "Resource": ["arn:aws:s3:::<BUCKET-NAME>/*"]
    }
]
```

Managing IAM Policies

IAM gives you the tools to create and manage all types of IAM policies (managed policies and inline policies). To add permissions to an IAM principal entity—that is, an IAM user, group, or role—you create a policy and then attach the policy to the principal entity. You can attach multiple policies to a principal entity, and each policy can contain multiple permissions.

Consult these resources for details:

- For more information about the different types of IAM policies, see [IAM Policies \(p. 275\)](#).
- For general information about using policies within IAM, see [Access Management \(p. 274\)](#).
- For information about how permissions are evaluated when multiple policies are in effect for a given IAM principal entity, see [IAM JSON Policy Evaluation Logic \(p. 458\)](#).
- For information about policy size and naming limitations, see [Limitations on IAM Entities and Objects \(p. 414\)](#).

Topics

- [Creating IAM Policies \(p. 317\)](#)
- [Validating JSON Policies \(p. 321\)](#)
- [Testing IAM Policies with the IAM Policy Simulator \(p. 322\)](#)
- [Attaching and Detaching IAM Policies \(p. 331\)](#)
- [Versioning IAM Policies \(p. 335\)](#)
- [Editing IAM Policies \(p. 337\)](#)
- [Deleting IAM Policies \(p. 341\)](#)
- [Reducing Policy Scope by Viewing User Activity \(p. 342\)](#)

Creating IAM Policies

A [policy \(p. 275\)](#) is an entity that, when attached to an identity or resource, defines their permissions. Policies are stored in AWS as JSON documents and are attached to principals as *identity-based policies* in IAM. You can attach an identity-based policy to a principal (or identity), such as an IAM group, user, or role. Identity-based policies include AWS managed policies, customer managed policies, and [inline policies \(p. 279\)](#).

You can create a new IAM policy in the AWS Management Console using one of the following methods:

- **Import** – You can import a managed policy within your account and then edit the policy to customize it to your specific requirements. A managed policy can be an AWS managed policy, or a customer managed policy that you created previously.
- **Visual editor** – You can construct a new policy from scratch in the visual editor. If you use the visual editor, you do not have to understand JSON syntax.
- **JSON** – In the **JSON** tab, you can create a policy using JSON syntax. You can type a new JSON policy document or paste an [example policy \(p. 295\)](#).

You can create an inline policy in the AWS Management Console. An inline policy is one that you create and embed directly to an IAM group, user, or role. To learn more, see [the section called "Attaching IAM Policies \(Console\)" \(p. 331\)](#). You cannot create AWS managed policies.

For information about policy size limitations and other quotas, see [Limitations on IAM Entities and Objects \(p. 414\)](#).

Topics

- [Create an IAM Policy \(Console\) \(p. 317\)](#)
- [Import an Existing Managed Policy \(p. 318\)](#)
- [Create a Policy with the Visual Editor \(p. 319\)](#)
- [Create a Policy on the JSON Tab \(p. 320\)](#)
- [Create IAM Policies \(AWS CLI or AWS API\) \(p. 321\)](#)

Create an IAM Policy (Console)

No matter which way you choose to create a policy, they all start the same way:

To start creating a new policy (console)

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. Choose **Create policy**.
4. Choose one of the following ways to create the policy. Then follow the steps in the corresponding procedure:
 - [Import an Existing Managed Policy \(p. 318\)](#)
 - [Create a Policy with the Visual Editor \(p. 319\)](#)
 - [Create a Policy on the JSON Tab \(p. 320\)](#)

Import an Existing Managed Policy

An easy way to create a new policy is to import an existing managed policy within your account that has at least some of the permissions that you need. You can then customize the policy to match it to your new requirements.

You cannot import an inline policy. To learn about the difference between managed and inline policies, see [Managed Policies and Inline Policies \(p. 279\)](#).

To import an existing managed policy in the visual editor

1. Start the **Create policy** wizard by following the steps in [Create an IAM Policy \(Console\) \(p. 317\)](#). Choose the **Visual editor** tab, and then on the right side of the page, choose **Import managed policy**.
 2. In the **Import managed policies** window, choose the managed policies that most closely match the policy that you want to include in your new policy. You can use the **Filter** menu or type in the search box at the top to limit the results in the list of policies.
 3. Choose **Import**.
- The imported policies are added in new permission blocks at the bottom of your policy.
4. Use the **Visual editor** or choose **JSON** to customize your policy. Then choose **Review policy**.

Note

You can switch between the **Visual editor** and **JSON** tabs any time. However, if you make changes or choose **Review policy** in the **Visual editor** tab, IAM might restructure your policy to optimize it for the visual editor. For more information, see [Policy Restructuring \(p. 387\)](#).

5. On the **Review** page, type a **Name** and a **Description** (optional) for the policy that you are creating. You cannot edit these fields later. Review the policy **Summary** and then choose **Create policy** to save your work.

To import an existing managed policy in the JSON tab

1. Start the **Create policy** wizard by following the steps in [Create an IAM Policy \(Console\) \(p. 317\)](#). Choose the **JSON** tab, and then on the right side of the page, choose **Import managed policy**.
 2. In the **Import managed policies** window, choose the managed policies that most closely match the policy that you want to include in your new policy. You can use the **Filter** menu or type in the search box at the top to limit the results in the list of policies.
 3. Choose **Import**.
- Statements from the imported policies are added to the bottom of your JSON policy.
4. Customize your policy in JSON, or choose the **Visual editor**. Then choose **Review policy**.

Note

You can switch between the **Visual editor** and **JSON** tabs any time. However, if you make changes or choose **Review policy** in the **Visual editor** tab, IAM might restructure your policy to optimize it for the visual editor. For more information, see [Policy Restructuring \(p. 387\)](#).

5. On the **Review policy** page, type a **Name** and a **Description** (optional) for the policy that you are creating. You cannot edit these later. Review the policy **Summary** and then choose **Create policy** to save your work.

After you create a policy, you can attach it to your groups, users, or roles. For more information, see [Attaching and Detaching IAM Policies \(p. 331\)](#).

Create a Policy with the Visual Editor

The visual editor in the IAM console guides you through creating a policy without having to write JSON syntax. To view an example of using the visual editor to create a policy, see [the section called "Controlling Access to Identities" \(p. 288\)](#).

To use the visual editor to create a policy

1. Start the **Create Policy** wizard by following the steps in [Create an IAM Policy \(Console\) \(p. 317\)](#).
2. On the **Visual editor** tab, choose **Choose a service**. Then choose an AWS service to add to the policy. You can use the search box at the top to limit the results in the list of services. You can choose only one service within a visual editor permission block. To grant access to more than one service, add multiple permission blocks by choosing **Add additional permissions**.
3. Choose **Select actions** and then choose the actions to add to the policy. The visual editor shows the actions available in the service that you selected in the previous step.

You can choose actions in the following ways:

- Use check boxes to select all actions for the service or all actions in one of the predefined **Access level** groups.
- Expand each of the **Access level** groups to choose individual actions.
- Choose **add actions** to type a specific action or use wildcards (*) to specify multiple actions.

By default, the policy that you are creating allows the actions that you choose. To deny the chosen actions instead, choose **Switch to deny permissions**. Because [IAM denies by default \(p. 458\)](#), we recommend as a security best practice that you allow permissions to only those actions and resources that a user needs. This is sometimes called "whitelisting." You should create a JSON statement to deny permissions ("blacklisting") only if you want to override a permission separately allowed by another statement or policy. We recommend that you limit the number of deny permissions to a minimum because they can increase the difficulty of troubleshooting permissions.

4. If the selected service and the actions that you selected in the previous steps do not support choosing [specific resources \(p. 294\)](#), then **All resources** is selected for you. In that case, you cannot edit this section.

If you chose one or more actions that support [resource-level permissions \(p. 294\)](#), then the visual editor lists those resources. You can then choose **Resources** to specify resources for your policy.

You can choose resources in the following ways:

- Choose **Add ARN** to provide the details about your resource. Instead of typing a value, you can also choose **Any** to provide permissions for any value for the specified setting. For example, if you selected the Amazon EC2 **Read** access level group, then the actions in your policy support the **instance** resource type. You must provide the **Region**, **Account**, and **InstanceId** values for your resource. If you provide your account ID but choose **Any** for the region and instance ID, then the policy grants permissions to any instance in your account.
 - Choose **Add ARN** to specify resources by their [Amazon Resource Name \(ARN\)](#). You can include a wildcard (*) in any field of the ARN (between each pair of colons). For more information, see [IAM JSON Policy Elements: Resource \(p. 436\)](#).
 - Choose **Any** from the far right of the resource section to grant permissions to any resources of a particular type.
 - Choose **All resources** to choose all resources for that service.
5. (Optional) Choose **Specify request conditions (optional)** to add conditions to the policy that you are creating. Conditions limit a JSON policy statement's effect. For example, you can specify that a user is allowed to perform the actions on the resources only when that user's request happens within a certain time range. You can also use commonly used conditions to limit whether a user

must be authenticated using a multi-factor authentication (MFA) device, or if the request must originate from within a certain range of IP addresses. For lists of all of the context keys that you can use in a policy condition, see [AWS Service Actions and Condition Context Keys for Use in IAM Policies \(p. 485\)](#).

You can choose conditions in the following ways:

- Use check boxes to select commonly used conditions.
- Choose **Add condition** to specify other conditions. Choose the condition's **Condition Key**, **Qualifier**, and **Operator**, and then type a **Value**. To add more than one value, choose **Add new value**. You can consider the values as being connected by a logical "OR" operator. When you are finished, choose **Add**.

To add more than one condition, choose **Add condition** again. Repeat as needed. Each condition applies only to this one visual editor permission block. All the conditions must be true for the permission block to be considered a match. In other words, consider the conditions to be connected by a logical "AND" operator.

For more information about the **Condition** element, see [IAM JSON Policy Elements: Condition \(p. 438\)](#) in the [IAM JSON Policy Reference \(p. 425\)](#).

6. To add more permission blocks, choose **Add additional permissions**. For each block, repeat steps 2 through 5.
7. When you are finished, choose **Review policy**.

Note

You can switch between the **Visual editor** and **JSON** tabs any time. However, if you make changes or choose **Review policy** in the **Visual editor** tab, IAM might restructure your policy to optimize it for the visual editor. For more information, see [Policy Restructuring \(p. 387\)](#).

8. On the **Review policy** page, type a **Name** and a **Description** (optional) for the policy that you are creating. Review the policy summary to make sure that you have granted the intended permissions, and then choose **Create policy** to save your new policy.

After you create a policy, you can attach it to your groups, users, or roles. For more information, see [Attaching and Detaching IAM Policies \(p. 331\)](#).

Create a Policy on the JSON Tab

You can type or paste policies in JSON by choosing the **JSON** tab. This method is useful for copying an [example policy \(p. 295\)](#) to use in your account. Or, you can type your own JSON policy document in the JSON editor. You can also use the **JSON** tab to toggle between the visual editor and JSON to compare the views.

A JSON [policy \(p. 275\)](#) document consists of one or more statements. Each statement should contain all the actions that share the same effect (Allow or Deny) and support the same resources and conditions. If one action requires you to specify all resources ("*") and another action supports the [Amazon Resource Name \(ARN\)](#) of a specific resource, they must be in two separate JSON statements. For general information about IAM policies, see [IAM Policies \(p. 275\)](#). For information about the IAM policy language, see [IAM JSON Policy Reference \(p. 425\)](#).

To use the JSON policy editor to create a policy

1. Start the **Create Policy** wizard by following the steps in [Create an IAM Policy \(Console\) \(p. 317\)](#).
2. Choose the **JSON** tab.
3. Type or paste a JSON policy document. For details about the IAM policy language, see [IAM JSON Policy Reference \(p. 425\)](#).

4. When you are finished, choose **Review policy**. The [Policy Validator \(p. 321\)](#) reports any syntax errors.

Note

You can switch between the **Visual editor** and **JSON** tabs any time. However, if you make changes or choose **Review policy** in the **Visual editor** tab, IAM might restructure your policy to optimize it for the visual editor. For more information, see [Policy Restructuring \(p. 387\)](#).

5. On the **Review policy** page, type a **Name** and a **Description** (optional) for the policy that you are creating. Review the policy **Summary** to see the permissions that are granted by your policy. Then choose **Create policy** to save your work.

After you create a policy, you can attach it to your groups, users, or roles. For more information, see [Attaching and Detaching IAM Policies \(p. 331\)](#).

Create IAM Policies (AWS CLI or AWS API)

You can create an IAM policy or an inline policy using the AWS Command Line Interface (AWS CLI) or the AWS API.

To create a customer managed policy (AWS CLI or API)

- AWS CLI: [create-policy](#)
- IAM API: [CreatePolicy](#)

To create an inline policy for a principal entity (group, user, or role) (AWS CLI or API)

- AWS CLI:
 - [put-group-policy](#)
 - [put-role-policy](#)
 - [put-user-policy](#)
- IAM API:
 - [PutGroupPolicy](#)
 - [PutRolePolicy](#)
 - [PutUserPolicy](#)

Note

You can embed an inline policy for a [service-linked role \(p. 136\)](#) only in the service that depends on the role. See the [AWS documentation](#) for your service to see whether it supports this feature.

Validating JSON Policies

Policy Validator automatically examines your new and existing IAM access control policies to ensure that they comply with the IAM policy grammar. A **policy** is a JSON document written using the [IAM policy grammar](#). It defines access permissions for the AWS user, group, or role that you attach the policy to. If Policy Validator determines that a policy is not in compliance with the grammar, it prompts you to fix the policy. Policy Validator is only available if you have non-compliant policies.

You can use the following methods to access Policy Validator:

1. **Creating JSON policies** – When you create a new JSON policy, Policy Validator runs automatically when you choose **Review policy**. If the policy is not valid, you receive a notification and must fix the problem before you can continue.

2. **Editing JSON policies** – When you edit an existing JSON policy, Policy Validator runs automatically when you choose **Review policy**. If the policy is not valid, you receive a notification and must fix the problem before you can continue. Existing policies with errors that were set up before the introduction of Policy Validator will continue to function. However, you cannot edit and save them without fixing the policy syntax errors.

Note

Policy Validator only checks JSON policy syntax and grammar. It does not validate that your ARNs, action names, or condition keys are correct.

Testing IAM Policies with the IAM Policy Simulator

For more information about how and why to use IAM policies, see [IAM Policies \(p. 275\)](#).

You can access the **IAM Policy Simulator Console** at: <https://pollicysim.aws.amazon.com/>

The following video provides an introduction to using the IAM policy simulator.

[Getting Started with the IAM Policy Simulator](#)

With the IAM policy simulator, you can test and troubleshoot IAM and resource-based policies in the following ways:

- Test policies that are attached to IAM users, groups, or roles in your AWS account. If more than one policy is attached to the user, group, or role, you can test all the policies, or select individual policies to test. You can test which actions are allowed or denied by the selected policies for specific resources.
- Test policies that are attached to AWS resources, such as Amazon S3 buckets, Amazon SQS queues, Amazon SNS topics, or Amazon Glacier vaults.
- If your AWS account is a member of an [AWS Organization](#), then you can test the impact of organization control policies on your IAM policies and resource policies.
- Test new policies that are not yet attached to a user, group, or role by typing or copying them into the simulator. These are used only in the simulation and are not saved. Note: you cannot type or copy a resource-based policy into the simulator. To use a resource-based policy in the simulator, you must include the resource in the simulation and select the check box to include that resource's policy in the simulation.
- Test the policies with selected services, actions, and resources. For example, you can test to ensure that your policy allows an entity to perform the `ListAllMyBuckets`, `CreateBucket`, and `DeleteBucket` actions in the Amazon S3 service on a specific bucket.
- Simulate real-world scenarios by providing context keys, such as an IP address or date, that are included in Condition elements in the policies being tested.
- Identify which specific statement in a policy results in allowing or denying access to a particular resource or action.

Topics

- [How the IAM Policy Simulator Works \(p. 322\)](#)
- [Permissions Required for Using the IAM Policy Simulator \(p. 323\)](#)
- [Using the IAM Policy Simulator \(AWS Management Console\) \(p. 325\)](#)

How the IAM Policy Simulator Works

The simulator evaluates the policies that you choose and determines the effective permissions for each of the actions that you specify. The simulator uses the same policy evaluation engine that is used during

real requests to AWS services. But the simulator differs from the live AWS environment in the following ways:

- The simulator does not make an actual AWS service request, so you can safely test requests that might make unwanted changes to your live AWS environment.
- Because the simulator does not simulate running the selected actions it cannot report any response to the simulated request. The only result returned is whether the requested action would be allowed or denied.
- If you edit a policy inside the simulator, these changes affect only the simulator. The corresponding policy in your AWS account remains unchanged.

Permissions Required for Using the IAM Policy Simulator

You can use the policy simulator console or the policy simulator API to test policies. By default, console users can test policies that are not yet attached to a user, group, or role by typing or copying those policies into the simulator. These policies are used only in the simulation and do not disclose sensitive information. API users must have permissions to test unattached policies. To allow console or API users to test policies that are attached to IAM users, groups, or roles in your AWS account, you must provide users with permissions to retrieve those policies. In order to test resource-based policies, users must have permission to retrieve the resource's policy.

For examples of console and API policies that allow a user to simulate policies, see [the section called "Example Policies: AWS Identity and Access Management \(IAM\)" \(p. 296\)](#).

Permissions Required for Using the Policy Simulator Console

To allow users to test policies that are attached to IAM users, groups, or roles in your AWS account, you must provide your users with permissions to retrieve those policies. In order to test resource-based policies, users must have permission to retrieve the resource's policy.

To view an example policy that allows using the policy simulator console for policies that are attached to a user, group, or role, see [IAM: Access the Policy Simulator Console \(p. 306\)](#).

To view an example policy that allows using the policy simulator console only for those users with a specific path, see [IAM: Access the Policy Simulator Console Based on User Path \(p. 307\)](#).

To create a policy to allow using the policy simulator console for only one type of entity, use the following procedures.

To allow console users to simulate policies for users

Include the following actions in your policy:

- `iam:GetGroupPolicy`
- `iam:GetPolicy`
- `iam:GetPolicyVersion`
- `iam:GetUser`
- `iam:GetUserPolicy`
- `iam>ListAttachedUserPolicies`
- `iam>ListGroupsForUser`
- `iam>ListGroupPolicies`
- `iam>ListUserPolicies`

- iam>ListUsers

To allow console users to simulate policies for groups

Include the following actions in your policy:

- iam>GetGroup
- iam>GetGroupPolicy
- iam>GetPolicy
- iam>GetPolicyVersion
- iam>ListAttachedGroupPolicies
- iam>ListGroupPolicies
- iam>ListGroups

To allow console users to simulate policies for roles

Include the following actions in your policy:

- iam>GetPolicy
- iam>GetPolicyVersion
- iam>GetRole
- iam>GetRolePolicy
- iam>ListAttachedRolePolicies
- iam>ListRolePolicies
- iam>ListRoles

To test resource-based policies, users must have permission to retrieve the resource's policy.

To allow console users to test resource-based policies in an Amazon S3 bucket

Include the following actions in your policy:

- s3>GetBucketPolicy
- s3>GetObject

For example, the following policy uses these actions to allow console users to simulate a resource-based policy in a specific Amazon S3 bucket.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "s3:GetBucketPolicy",  
                "s3:GetObject"  
            ],  
            "Effect": "Allow",  
            "Resource": "arn:aws:s3:::<BUCKET-NAME>/*"  
        }  
    ]  
}
```

To allow console users to test policies for AWS Organizations

Include the following actions in your policy:

- organizations:DescribePolicy
- organizations>ListPolicies
- organizations>ListPoliciesForTarget
- organizations>ListTargetsForPolicy

Permissions Required for Using the Policy Simulator API

The policy simulator API actions [GetContextKeyForCustomPolicy](#) and [SimulateCustomPolicy](#) allow users to test policies that are not yet attached to a user, group, or role by passing the policies as strings to the API. These policies are used only in the simulation and do not disclose sensitive information. To allow API users to test policies that are attached to IAM users, groups, or roles in your AWS account, you must provide users with permissions to call [GetContextKeyForPrincipalPolicy](#) and [SimulatePrincipalPolicy](#).

To view an example policy that allows using the policy simulator API for unattached policies and policies attached to a user, group, or role in the current AWS account, see [IAM: Access the Policy Simulator API \(p. 306\)](#).

To create a policy to allow using the policy simulator API for only one type of policy, use the following procedures.

To allow API users to simulate policies passed directly to the API as strings

Include the following actions in your policy:

- iam:GetContextKeysForCustomPolicy
- iam:SimulateCustomPolicy

To allow API users to simulate policies attached to IAM users, groups, roles, or resources

Include the following actions in your policy:

- iam:GetContextKeysForPrincipalPolicy
- iam:SimulatePrincipalPolicy

For example, to give a user named Bob permission to simulate a policy that is assigned to a user named Alice, give Bob access to the following resource: `arn:aws:iam::777788889999:user/alice`.

To view an example policy that allows using the policy simulator API only for those users with a specific path, see [IAM: Access the Policy Simulator API Based on User Path \(p. 307\)](#).

Using the IAM Policy Simulator (AWS Management Console)

By default, users can test policies that are not yet attached to a user, group, or role by typing or copying those policies into the policy simulator console. These policies are used only in the simulation and do not disclose sensitive information.

To test a policy that is not attached to a user, group, or role using the policy simulator console:

1. Open the IAM policy simulator console at: <https://pollicysim.aws.amazon.com/>.
2. In the **Mode:** menu at the top of the page, choose **New Policy**.

3. In the **Policy Sandbox**, choose **Create New Policy**.
4. Type or copy a policy into the simulator, and use the simulator as described in the following steps.

After you have been given the required permissions for using the IAM Policy Simulator Console, you can use the simulator to test an IAM user, group, role, or resource policy.

To use the policy simulator console:

1. Open the IAM policy simulator console at <https://policysim.aws.amazon.com/>.

Note

To sign in to the policy simulator as an IAM user, use your unique sign-in URL to sign in to the AWS Management Console. Then go to <https://policysim.aws.amazon.com/>. For more information about signing in as an IAM user, see [How IAM Users Sign In to AWS \(p. 67\)](#).

The simulator opens in **Existing Policies** mode and lists the IAM users in your account under **Users, Groups, and Roles**.

2. Choose the option that is appropriate to your task:

To test this:	Do this:
A policy attached to a user	Choose Users in the Users, Groups, and Roles list. Then choose the user.
A policy attached to a group	Choose Groups in the Users, Groups, and Roles list. Then choose the group.
A policy attached to a role	Choose Roles in the Users, Groups, and Roles list. Then choose the role.
A policy attached to a resource	See Step 8 .
A custom policy	Choose New Policy from the mode list at the top. Then, in the Policy Sandbox pane on the left, choose Create New Policy , type or paste a policy, then choose Apply .

Tip

To test a policy that is attached to group, you can launch the IAM policy simulator directly from the [IAM console](#): In the navigation pane, choose **Groups**. Choose the name of the group that you want to test a policy on, and then choose the **Permissions** tab. In the **Inline Policies** or **Managed Policies** section, locate the policy that you want to test. In the **Actions** column for that policy, choose **Simulate Policy**.

To test a customer managed policy that is attached to a user: In the navigation pane, choose **Users**. Choose the name of the user that you want to test a policy on. Then choose the **Permissions** tab and expand the policy that you want to test. On the far right, choose **Simulate policy**. The **IAM Policy Simulator** opens in a new window and displays the selected policy in the **Policies** pane.

3. (Optional) If your account is a member of an [AWS Organization](#), then any organization control policies (OCPs) that affect the simulated user's account appear in the **Policies** pane along with **IAM policies** and **Resource policies**. These policies are essentially filters that restrict what permissions can be used by users, groups, or roles in an affected account. If an OCP blocks a service or action, then no entity in that account can access that service nor perform that action. This is true even if an administrator explicitly grants permissions to that service or action through an IAM or resource policy. To remove an OCP from the simulation, clear the check box next to the OCP name. To view the OCP contents, choose the name of the OCP.

- If your account is not a member of an organization, then there are no OCPs to simulate.
4. (Optional) To test only a subset of policies attached to a user, group, or role, in the **Policies** pane clear the check box next to each policy that you want to exclude.
 5. Under **Policy Simulator**, choose **Select service** and then choose the service to test. Then choose **Select actions** and select one or more actions to test. Although the menus show the available selections for only one service at a time, all the services and actions that you have selected appear in **Action Settings and Results**.
 6. (Optional) If any of the policies that you choose in **Step 2** and **Step 4** include conditions with [AWS global condition keys \(p. 476\)](#), then supply values for those keys. You can do this by expanding the **Global Settings** section and typing values for the key names displayed there.

Warning

If you leave the value for a condition key empty, then that key is ignored during the simulation. In some cases, this results in an error and the simulation fails to run. In other cases the simulation runs, but the results might not be reliable because the simulation does not match the real-world conditions that include a value for the condition key or variable.

7. (Optional) Each selected action appears in the **Action Settings and Results** list with **Not simulated** shown in the **Permission** column until you actually run the simulation. Before you run the simulation, you can configure each action with a resource. To configure individual actions for a specific scenario, choose the arrow to expand the action's row. If the action supports resource-level permissions, you can type the [Amazon Resource Name \(ARN\)](#) of the specific resource whose access you want to test. By default, each resource is set to a wildcard (*). You can also specify a value for any [condition context keys \(p. 485\)](#). As noted previously, keys with empty values are ignored, which can cause simulation failures or unreliable results.
 - a. Choose the arrow next to the action name to expand each row and configure any additional information required to accurately simulate the action in your scenario. If the action requires any resource-level permissions, you can type the [Amazon Resource Name \(ARN\)](#) of the specific resource that you want to simulate access to. By default, each resource is set to a wildcard (*).
 - b. If the action supports resource-level permissions but does not require them, then you can choose **Add Resource** to select the resource type that you want to add to the simulation.
 - c. If any of the selected policies include a Condition element that references a context key for this action's service, then that key name is displayed under the action. You can specify the value to be used during the simulation of that action for the specified resource.

Actions that require different groups of resource types

Some actions require different resource types under different circumstances. Each group of resource types is associated with a scenario. If one of these applies to your simulation, select it and the simulator requires the resource types appropriate for that scenario. The following list shows each of the supported scenario options and the resources that you must define to run the simulation.

Each of the following Amazon EC2 scenarios requires that you specify instance, image, and security-group resources. If your scenario includes an EBS volume, then you must specify that volume as a resource. If the Amazon EC2 scenario includes a virtual private cloud (VPC), then you must supply the network-interface resource. If it includes an IP subnet, then you must specify the subnet resource. For more information on the Amazon EC2 scenario options, see [Supported Platforms](#) in the *AWS EC2 User Guide*.

- **EC2-Classic-InstanceStore**

instance, image, security-group

- **EC2-Classic-EBS**

instance, image, security-group, volume

- **EC2-VPC-InstanceStore**
instance, image, security-group, network-interface
 - **EC2-VPC-InstanceStore-Subnet**
instance, image, security-group, network-interface, subnet
 - **EC2-VPC-EBS**
instance, image, security-group, network-interface, volume
 - **EC2-VPC-EBS-Subnet**
instance, image, security-group, network-interface, subnet, volume
8. (Optional) If you want to include a resource-based policy in your simulation, then you must first select the actions that you want to simulate on that resource in [Step 5](#). Expand the rows for the selected actions, and type the ARN of the resource with a policy that you want to simulate. Then select **Include Resource Policy** next to the **ARN** text box. The IAM policy simulator currently supports resource-based policies from only the following services: Amazon S3 (resource-based policies only; ACLs are not currently supported), Amazon SQS, Amazon SNS, and unlocked Amazon Glacier vaults (locked vaults are not currently supported).
 9. Choose **Run Simulation** in the upper-right corner.

The **Permission** column in each row of **Action Settings and Results** displays the result of the simulation of that action on the specified resource.

10. To see which statement in a policy explicitly allowed or denied an action, choose the [N matching statement\(s\)](#) link in the **Permissions** column to expand the row. Then choose the [Show statement](#) link. The **Policies** pane shows the relevant policy with the statement that affected the simulation result highlighted.

Note

If an action is *implicitly* denied—that is, if the action is denied only because it is not explicitly allowed—the **List** and **Show statement** options are not displayed.

Troubleshooting IAM Policy Simulator Console Messages

The following table lists the informational and warning messages you might encounter when using the IAM policy simulator. The table also provides steps you can take to resolve them.

Message	Steps to resolve
This policy has been edited. Changes will not be saved to your account.	<p>No action required.</p> <p>This message is informational. If you edit an existing policy in the IAM policy simulator, your change does not affect your AWS account. The simulator allows you to make changes to policies for testing purposes only.</p>
Cannot get the resource policy. Reason: detailed error message	<p>The simulator is not able to access a requested resource-based policy. Ensure that the specified resource ARN is correct and that the user running the simulation has permission to read the resource's policy.</p>
One or more policies require values in the simulation settings. The simulation might fail without these values.	<p>This message appears if the policy you are testing contains condition keys or variables but you have</p>

Message	Steps to resolve
	<p>not entered any values for these keys or variables in Simulation Settings.</p> <p>To dismiss this message, choose Simulation Settings, Then type a value for each condition key or variable.</p>
You have changed policies. These results are no longer valid.	<p>This message appears if you have changed the selected policy while results are displayed in the Results pane. Results shown in the Results pane are not updated dynamically.</p> <p>To dismiss this message, choose Run Simulation again to display new simulation results based on the changes made in the Policies pane.</p>
The resource you entered for this simulation does not match this service.	<p>This message appears if you have entered an Amazon Resource Name (ARN) in the Simulation Settings pane that does not match the service that you chose for the current simulation. For example, this message appears if you specify an ARN for an Amazon DynamoDB resource but you chose Amazon Redshift as the service to simulate.</p> <p>To dismiss this message, do one of the following:</p> <ul style="list-style-type: none"> Remove the ARN from the box in the Simulation Settings pane. Choose the service that matches the ARN that you specified in Simulation Settings.
This action belongs to a service that supports special access control mechanisms in addition to resource-based policies, such as S3 ACLs or Glacier vault lock policies. The policy simulator does not support these mechanisms, so the results can differ from your production environment.	<p>No action required.</p> <p>This message is informational. In the current version, the simulator evaluates policies attached to users and groups, and can evaluate resource-based policies for Amazon S3, Amazon SQS, Amazon SNS, and Amazon Glacier. The policy simulator does not support all access control mechanisms supported by other AWS services.</p>
DynamoDB FGAC is currently not supported.	<p>No action required.</p> <p>This informationl message refers to <i>fine-grained access control</i>. This is the ability to use IAM policy conditions to determine who can access individual data items and attributes in DynamoDB tables and indexes as well as the actions that can be performed on them. The current version of the IAM policy simulator does not support this type of policy condition. For more information on DynamoDB fine-grained access control, see Fine-Grained Access Control for DynamoDB.</p>

Message	Steps to resolve
You have policies that do not comply with the policy syntax. You can use the Policy Validator to review and accept the recommended updates to your policies.	This message appears at the top of the policy list if you have policies that do not comply with the IAM policy grammar. In order to simulate these policies, follow the instructions at Validating JSON Policies (p. 321) to identify and fix these policies.
This policy must be updated to comply with the latest policy syntax rules.	This message is displayed if you have policies that do not comply with the IAM policy grammar. In order to simulate these policies, follow the instructions at Validating JSON Policies (p. 321) to identify and fix these policies.

Using the IAM Policy Simulator (AWS CLI, Tools for Windows PowerShell, and AWS API)

Policy simulator commands typically require calling APIs to do two things:

1. Evaluate the policies and return the list of context keys that they reference. You need to know what context keys are referenced so that you can supply values for them in the next step.
2. Simulate the policies, providing a list of actions, resources, and context keys that are used during the simulation.

For security reasons, the APIs have been broken into two groups:

- API actions that simulate only policies that are passed directly to the API as strings. This set includes [GetContextKeysForCustomPolicy](#) and [SimulateCustomPolicy](#).
- API actions that simulate the policies that are attached to a specified IAM user, group, role, or resource. Because these APIs can reveal details of permissions assigned to other IAM entities, you should consider restricting access to these API actions. This set includes [GetContextKeysForPrincipalPolicy](#) and [SimulatePrincipalPolicy](#). For more information about restricting access to API actions, see [Example Policies: AWS Identity and Access Management \(IAM\) \(p. 296\)](#).

In both cases, the API actions simulate the effect of one or more policies on a list of actions and resources. Each action is paired with each resource and the simulation determines whether the policies allow or deny that action for that resource. You can also provide values for any context keys that your policies reference. You can get the list of context keys that the policies reference by first calling [GetContextKeysForCustomPolicy](#) or [GetContextKeysForPrincipalPolicy](#). If you don't provide a value for a context key, the simulation still runs. But the results might not be reliable because the simulator cannot include that context key in the evaluation.

To get the list of context keys

Use these commands to evaluate a list of policies and return a list of context keys that are used in the policies.

- AWS CLI: `aws iam get-context-keys-for-custom-policy` and `aws iam get-context-keys-for-principal-policy`
- Tools for Windows PowerShell: [Get-IAMContextKeysForCustomPolicy](#) and [Get-IAMContextKeysForPrincipalPolicy](#)
- AWS API: [GetContextKeysForCustomPolicy](#) and [GetContextKeysForPrincipalPolicy](#)

To simulate IAM policies

Use these commands to simulate IAM policies to determine a user's effective permissions.

- AWS CLI: `aws iam simulate-custom-policy` and `aws iam simulate-principal-policy`
- Tools for Windows PowerShell: `Test-IAMCustomPolicy` and `Test-IAMPrincipalPolicy`
- AWS API: `SimulateCustomPolicy` and `SimulatePrincipalPolicy`

Attaching and Detaching IAM Policies

You can attach and detach managed policies and embed and delete inline policies using the AWS Management Console, the AWS Command Line Interface (AWS CLI), or the AWS API.

For more information about the difference between managed and inline policies, see [Managed Policies and Inline Policies \(p. 279\)](#).

For general information about IAM policies, see [IAM Policies \(p. 275\)](#).

For information about policy size limitations and other quotas, see [Limitations on IAM Entities and Objects \(p. 414\)](#).

Topics

- [Attaching IAM Policies \(Console\) \(p. 331\)](#)
- [Embedding Inline Policies \(Console\) \(p. 331\)](#)
- [Detaching IAM Policies \(Console\) \(p. 332\)](#)
- [Attaching or Detaching IAM Policies \(AWS CLI or AWS API\) \(p. 333\)](#)

Attaching IAM Policies (Console)

You can use the AWS Management Console to attach a managed policy to an identity (a user, group, or role). Attaching a policy applies the permissions in the policy to the identity.

To attach a managed policy (console)

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, choose **Policies**.
3. In the list of policies, select the check box next to the name of the policy to attach. You can use the **Filter** menu and the search box to filter the list of policies.
4. Choose **Policy actions**, and then choose **Attach**.
5. Select one or more identities to attach the policy to. You can use the **Filter** menu and the search box to filter the list of principal entities. After selecting the identities, choose **Attach policy**.

Embedding Inline Policies (Console)

You can use the AWS Management Console to embed an inline policy in an identity (a user, group, or role). Embedding a policy applies the permissions in the policy to the identity. Because an inline policy is stored in the identity, it is embedded rather than attached, though the concept is similar.

To embed an inline policy for a user or role (console)

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, choose **Users or Roles**.
3. In the list, choose the name of the user or role to embed a policy in.
4. Choose the **Permissions** tab.
5. Scroll to the bottom of the page and choose **Add inline policy**.

Note

You cannot embed an inline policy in a [service-linked role \(p. 136\)](#) in IAM. Because the linked service defines whether you can modify the permissions of the role, you might be able to add additional policies from the service console, API, or AWS CLI. To view the service-linked role documentation for a service, see [AWS Services That Work with IAM \(p. 417\)](#) and choose **Yes** in the **Service-Linked Role** column for your service.

6. Choose from the following methods to view the steps required to create your policy:
 - [Import an Existing Managed Policy \(p. 318\)](#) – You can import a managed policy within your account and then edit the policy to customize it to your specific requirements. A managed policy can be an AWS managed policy or a customer managed policy that you created previously.
 - [Create a Policy with the Visual Editor \(p. 319\)](#) – You can construct a new policy from scratch in the visual editor. If you use the visual editor, you do not have to understand JSON syntax.
 - [Create a Policy on the JSON Tab \(p. 320\)](#) – In the **JSON** tab, you can use JSON syntax to create a policy. You can type a new JSON policy document or paste an [example policy \(p. 295\)](#).
7. After you create an inline policy, it is automatically embedded in your user or role.

To embed an inline policy for a group (console)

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, choose **Groups**.
3. In the list, choose the name of the group to embed a policy in.
4. Choose the **Permissions** tab and expand the **Inline Policies** section if necessary.
5. Choose **Create Group Policy**. If there are no existing policies in **Groups**, instead choose **click here** to create your first inline policy.
6. Choose **Policy Generator** or **Custom Policy**, and then choose **Select**.
7. Do one of the following:
 - If you chose **Custom Policy**, specify a name for the policy and create your policy document. [Policy Validator \(p. 321\)](#) reports any syntax errors.
 - If you are using the policy generator to create your policy, select the appropriate **Effect**, **AWS Service**, and **Actions** options. Type the Amazon Resource Name (ARN) (if applicable), and add any conditions that you want to include. Then choose **Add Statement**. You can add as many statements as you want to the policy. When you are finished adding statements, choose **Next Step**.
8. When you are satisfied with the policy, choose **Apply Policy**.

Detaching IAM Policies (Console)

You can use the AWS Management Console to detach a managed policy from a principal entity (a user, group, or role). Detaching a policy removes its permissions from the principal entity.

To detach a managed policy (Console)

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, choose **Policies**.
3. In the list of policies, select the check box next to the name of the policy to detach. You can use the **Filter** menu and the search box to filter the list of policies.
4. Choose **Policy actions**, and then choose **Detach**.
5. Select the identities to detach the policy from. You can use the **Filter** menu and the search box to filter the list of identities. After selecting the identities, choose **Detach policy**.

To detach and delete an inline policy (console)

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, choose **Groups, Users, or Roles**.
3. In the list, choose the name of the group, user, or role that has the policy you want to remove.
4. Choose the **Permissions** tab. If you chose **Groups**, expand the **Inline Policies** section if necessary.
5. If in **Groups**, choose **Remove Policy**. If in **Users or Roles**, choose X.

Attaching or Detaching IAM Policies (AWS CLI or AWS API)

You can attach or detach managed policies and embed or delete inline policies using the AWS Command Line Interface (AWS CLI) or the AWS API. The following information applies to both managed and inline policies.

To list managed policies (AWS CLI or API)

- AWS CLI: [list-policies](#)
- AWS API: [ListPolicies](#)

To retrieve detailed information about a managed policy (AWS CLI or API)

- AWS CLI: [get-policy](#)
- AWS API: [GetPolicy](#)

To list the identities (users, groups, and roles) to which a managed policy is attached (AWS CLI or API)

- AWS CLI: [list-entities-for-policy](#)
- AWS API: [ListEntitiesForPolicy](#)

To list the managed policies attached to an identity (a user, group, or role) (AWS CLI or API)

- AWS CLI:
 - [list-attached-group-policies](#)
 - [list-attached-role-policies](#)
 - [list-attached-user-policies](#)
- AWS API:
 - [ListAttachedGroupPolicies](#)
 - [ListAttachedRolePolicies](#)
 - [ListAttachedUserPolicies](#)

To list all inline policies that are attached to an identity (user, group, or role) (AWS CLI or API)

- AWS CLI:
 - [list-group-policies](#)
 - [list-role-policies](#)
 - [list-user-policies](#)
- AWS API:
 - [ListGroupPolicies](#)
 - [ListRolePolicies](#)
 - [ListUserPolicies](#)

To retrieve an inline policy document that is embedded in an identity (user, group, or role) (AWS CLI or API)

- AWS CLI:
 - [get-group-policy](#)
 - [get-role-policy](#)
 - [get-user-policy](#)
- AWS API:
 - [GetGroupPolicy](#)
 - [GetRolePolicy](#)
 - [GetUserPolicy](#)

To attach a managed policy to an identity (user, group, or role) (AWS CLI or API)

- AWS CLI:
 - [attach-group-policy](#)
 - [attach-role-policy](#)
 - [attach-user-policy](#)
- AWS API:
 - [AttachGroupPolicy](#)
 - [AttachRolePolicy](#)
 - [AttachUserPolicy](#)

To detach a managed policy from an identity (user, group, or role) (AWS CLI or API)

- AWS CLI:
 - [detach-group-policy](#)
 - [detach-role-policy](#)
 - [detach-user-policy](#)
- AWS API:
 - [DetachGroupPolicy](#)
 - [DetachRolePolicy](#)
 - [DetachUserPolicy](#)

To embed an inline policy in an identity (user, group, or role) (AWS CLI or API)

- AWS CLI:
 - [put-group-policy](#)
 - [put-role-policy](#)
 - [put-user-policy](#)
- AWS API:
 - [PutGroupPolicy](#)
 - [PutRolePolicy](#)
 - [PutUserPolicy](#)

Note

You can embed an inline policy for a [service-linked role \(p. 136\)](#) only in the service that depends on the role. See the [AWS documentation](#) for your service to see whether it supports this feature.

To detach and delete an inline policy from an identity (user, group, or role)

- AWS CLI:
 - [delete-group-policy](#)
 - [delete-role-policy](#)
 - [delete-user-policy](#)

Note

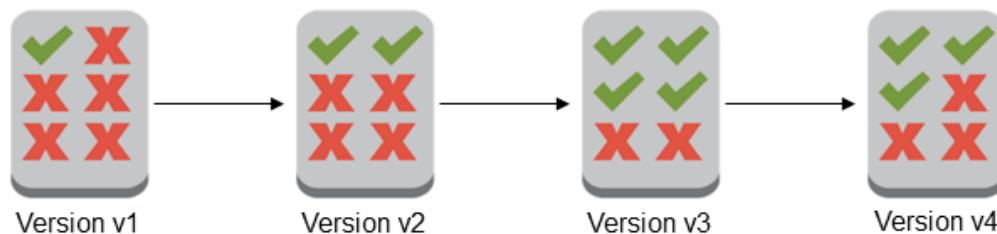
You can delete an inline policy from a [service-linked role \(p. 136\)](#) only in the service that depends on the role. See the [AWS documentation](#) for your service to see whether it supports this feature.

Versioning IAM Policies

When you make changes to an IAM customer managed policy, and when AWS makes changes to an AWS managed policy, the changed policy doesn't overwrite the existing policy. Instead, IAM creates a new *version* of the managed policy. IAM stores up to five versions of your customer managed policies. IAM does not support versioning for inline policies.

The following diagram illustrates versioning for a customer managed policy.

Multiple versions of a single managed policy



A policy version is different from a `Version` policy element. The `Version` policy element is used within a policy and defines the version of the policy language. To learn more about the `Version` policy element see [IAM JSON Policy Elements: Version \(p. 426\)](#).

You can use versions to track changes to a managed policy. For example, you might make a change to a managed policy and then discover that the change had unintended effects. In this case, you can roll back to a previous version of the managed policy by setting the previous version as the *default* version.

The following sections explain how you can use versioning for managed policies.

Topics

- [Setting the Default Version of Customer Managed Policies \(p. 336\)](#)
- [Using Versions to Roll Back Changes \(p. 337\)](#)
- [Version Limits \(p. 337\)](#)

Setting the Default Version of Customer Managed Policies

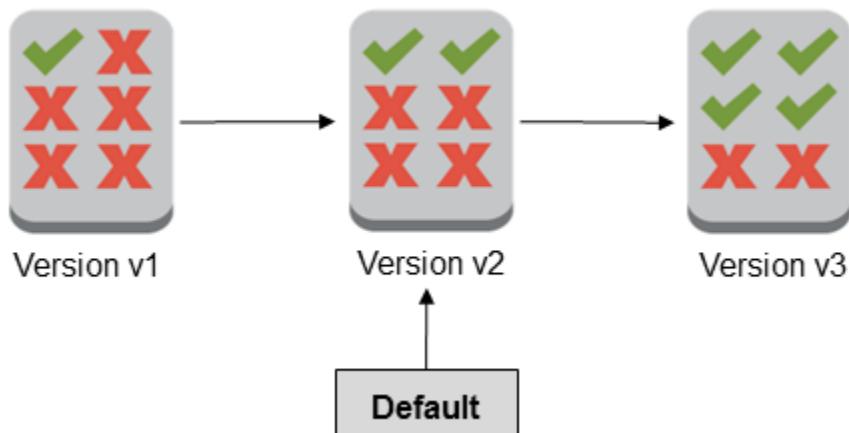
One of the versions of a managed policy is set as the *default* version. The policy's default version is the operative version—that is, it's the version that is in effect for all of the principal entities (users, groups, and roles) that the managed policy is attached to.

When you create a customer managed policy, the policy begins with a single version identified as v1. For managed policies with only a single version, that version is automatically set as the default. For customer managed policies with more than one version, you choose which version to set as the default. For AWS managed policies, the default version is set by AWS. The following diagrams illustrate this concept.

Managed policy with one version



Managed policy with multiple versions



You can set the default version of a customer managed policy to apply that version to every principal entity (user, group, and role) that the policy is attached to. You cannot set the default version for an AWS managed policy or an inline policy.

To set the default version of a customer managed policy (console)

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, choose **Policies**.
3. In the list of policies, choose the policy name of the policy to set the default version of. You can use the **Filter** menu and the search box to filter the list of policies.
4. Choose the **Policy versions** tab. Select the check box next to the version that you want to set as the default version, and then choose **Set as default**.

To learn how to set the default version of a customer managed policy from the AWS Command Line Interface or the AWS API, see [Editing IAM Policies \(AWS CLI or the AWS API\) \(p. 340\)](#).

Using Versions to Roll Back Changes

You can set the default version of a customer managed policy to roll back your changes. For example, consider the following scenario:

You create a customer managed policy that allows users to administer a particular Amazon S3 bucket using the AWS Management Console. Upon creation, your customer managed policy has only one version, identified as v1, so that version is automatically set as the default. The policy works as intended.

Later, you update the policy to add permission to administer a second Amazon S3 bucket. IAM creates a new version of the policy, identified as v2, that contains your changes. You set version v2 as the default, and a short time later your users report that they lack permission to use the Amazon S3 console. In this case, you can roll back to version v1 of the policy, which you know works as intended. To do this, you set version v1 as the default version. Your users are now able to use the Amazon S3 console to administer the original bucket.

Later, after you determine the error in version v2 of the policy, you update the policy again to add permission to administer the second Amazon S3 bucket. IAM creates another new version of the policy, identified as v3. You set version v3 as the default, and this version works as intended. At this point, you delete version v2 of the policy.

Version Limits

A managed policy can have up to five versions. If you need to make changes to a managed policy beyond five versions from the AWS Command Line Interface, or the AWS API, you must first delete one or more existing versions. If you use the AWS Management Console, you do not have to delete a version before editing your policy. When you save a sixth version, a dialog box appears that prompts you to delete one or more nondefault versions of your policy. You can view the JSON policy document for each version to help you decide. For details about this dialog box, see [the section called "Editing IAM Policies" \(p. 337\)](#).

You can delete any version of the managed policy that you want, except for the default version. When you delete a version, the version identifiers for the remaining versions do not change. As a result, version identifiers might not be sequential. For example, if you delete versions v2 and v4 of a managed policy and add two new versions, the remaining version identifiers might be v1, v3, v5, v6, and v7.

Editing IAM Policies

A [policy \(p. 275\)](#) is an entity that, when attached to an identity or resource, defines their permissions. Policies are stored in AWS as JSON documents and are attached to principals as *identity-based policies* in IAM. You can attach an identity-based policy to a principal (or identity), such as an IAM group, user, or role. Identity-based policies include AWS managed policies, customer managed policies, and [inline](#)

policies (p. 279). You can edit customer managed policies and inline policies in IAM. AWS managed policies cannot be edited. For information about policy size limitations and other quotas, see [Limitations on IAM Entities and Objects \(p. 414\)](#).

Editing Customer Managed Policies (Console)

You edit customer managed policies to change the permissions that are defined in the policy. A customer managed policy can have up to five versions. This is important because if you make changes to a managed policy beyond five versions, the AWS Management Console prompts you to decide which version to delete. You can also change the default version or delete a version of a policy before you edit it to avoid being prompted. To learn more about versions, see [Versioning IAM Policies \(p. 335\)](#).

To edit a customer managed policy (console)

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, choose **Policies**.
3. In the list of policies, choose the policy name of the policy to edit. You can use the **Filter** menu and the search box to filter the list of policies.
4. Choose the **Permissions** tab, and then choose **Edit policy**.
5. Do one of the following:
 - Choose the **Visual editor** tab to change your policy without understanding JSON syntax. You can make changes to the service, actions, resources, or optional conditions for each permission block in your policy. You can also import a policy to add additional permissions to the bottom of your policy. When you are finished making changes, choose **Review policy** to continue.
 - Choose the **JSON** tab to modify your policy by typing or pasting text in the JSON text box. You can also import a policy to add additional permissions to the bottom of your policy. When you are finished making changes, choose **Review policy** to continue. [Policy Validator \(p. 321\)](#) reports any syntax errors.

Note

You can switch between the **Visual editor** and **JSON** tabs any time. However, if you make changes or choose **Review policy** in the **Visual editor** tab, IAM might restructure your policy to optimize it for the visual editor. For more information, see [Policy Restructuring \(p. 387\)](#).

6. On the **Review** page, review the policy **Summary** and then choose **Save changes** to save your work.
7. If the managed policy already has the maximum of five versions, choosing **Save** displays a dialog box. To save your new version, you must remove at least one older version. You cannot delete the default version. Choose from the following options:
 - **Remove oldest non-default policy version (version v# - created # days ago)** – Use this option to see which version will be deleted and when it was created. You can view the JSON policy document for all nondefault versions by choosing the second option, **Select versions to remove**.
 - **Select versions to remove** – Use this option to view the JSON policy document and choose one or more versions to delete.

After choosing the versions to remove, choose **Delete version and save** to save your new policy version.

To set the default version of a customer managed policy (console)

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, choose **Policies**.
3. In the list of policies, choose the policy name of the policy to set the default version of. You can use the **Filter** menu and the search box to filter the list of policies.
4. Choose the **Policy versions** tab. Select the check box next to the version that you want to set as the default version, and then choose **Set as default**.

To delete a version of a customer managed policy (console)

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, choose **Policies**.
3. Choose the name of the customer managed policy that has a version you want to delete. You can use the **Filter** menu and the search box to filter the list of policies.
4. Choose the **Policy versions** tab. Select the check box next to the version that you want to delete. Then choose **Delete**.
5. Confirm that you want to delete the version, and then choose **Delete**.

Editing Inline Policies (Console)

You can edit an inline policy using the AWS Management Console.

To edit an inline policy for a user or role (console)

1. In the navigation pane, choose **Users or Roles**.
2. Choose the name of the user or role with the policy that you want to modify. Then choose the **Permissions** tab and expand the policy.
3. To edit an inline policy, choose **Edit Policy**.
4. Do one of the following:
 - Choose the **Visual editor** tab to change your policy without understanding JSON syntax. You can make changes to the service, actions, resources, or optional conditions for each permission block in your policy. You can also import a policy to add additional permissions to the bottom of your policy. When you are finished making changes, choose **Review policy** to continue.
 - Choose the **JSON** tab to modify your policy by typing or pasting text in the JSON text box. You can also import a policy to add additional permissions to the bottom of your policy. When you are finished making changes, choose **Review policy** to continue. [Policy Validator \(p. 321\)](#) reports any syntax errors. To save your changes without affecting the currently attached entities, clear the check box for **Save as default version**.

Note

You can switch between the **Visual editor** and **JSON** tabs any time. However, if you make changes or choose **Review policy** in the **Visual editor** tab, IAM might restructure your policy to optimize it for the visual editor. For more information, see [Policy Restructuring \(p. 387\)](#).

5. On the **Review** page, review the policy **Summary** and then choose **Save changes** to save your work.

To edit an inline policy for a group (console)

1. In the navigation pane, choose **Groups**.
2. Choose the name of the group with the policy that you want to modify. Then choose the **Permissions** tab.

3. To edit an inline policy, choose **Edit Policy**.
4. After you have modified your JSON policy, choose **Save** to save your changes.

Editing IAM Policies (AWS CLI or the AWS API)

You can edit a managed or inline policy using the AWS Command Line Interface (AWS CLI) or the AWS API.

Note

A managed policy can have up to five versions. If you need to make changes to a customer managed policy beyond five versions from the AWS Command Line Interface or the AWS API, you must first delete one or more existing versions.

To list managed policies (AWS CLI or API)

- AWS CLI: [list-policies](#)
- AWS API: [ListPolicies](#)

To retrieve detailed information about a managed policy (AWS CLI or API)

- AWS CLI: [get-policy](#)
- AWS API: [GetPolicy](#)

To list the identities (users, groups, and roles) to which a managed policy is attached (AWS CLI or API)

- AWS CLI: [list-entities-for-policy](#)
- AWS API: [ListEntitiesForPolicy](#)

To list the managed policies attached to an identity (a user, group, or role) (AWS CLI or API)

- AWS CLI:
 - [list-attached-group-policies](#)
 - [list-attached-role-policies](#)
 - [list-attached-user-policies](#)
- AWS API:
 - [ListAttachedGroupPolicies](#)
 - [ListAttachedRolePolicies](#)
 - [ListAttachedUserPolicies](#)

To edit a customer managed policy (AWS CLI or API)

- AWS CLI: [create-policy-version](#)
- AWS API: [CreatePolicyVersion](#)

To set the default version of a customer managed policy (AWS CLI or API)

- AWS CLI: [set-default-policy-version](#)
- AWS API: [SetDefaultPolicyVersion](#)

To delete a version of a customer managed policy (AWS CLI or API)

- AWS CLI: [delete-policy-version](#)
- AWS API: [DeletePolicyVersion](#)

Deleting IAM Policies

You can delete IAM policies using the AWS Management Console, the AWS Command Line Interface (AWS CLI), or the IAM API.

For more information about the difference between managed and inline policies, see [Managed Policies and Inline Policies \(p. 279\)](#).

For general information about IAM policies, see [IAM Policies \(p. 275\)](#).

For information about policy size limitations and other quotas, see [Limitations on IAM Entities and Objects \(p. 414\)](#).

Deleting Customer Managed Policies (Console)

You can delete a customer managed policy to remove it from your AWS account. You cannot delete AWS managed policies.

To delete a customer managed policy (console)

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, choose **Policies**.
3. Select the check box next to the customer managed policy to delete. You can use the **Filter** menu and the search box to filter the list of policies.
4. Choose **Policy actions**, and then choose **Delete**.
5. Confirm that you want to delete the policy, and then choose **Delete**.

To delete an inline policy for a group, user, or role (console)

1. In the navigation pane, choose **Groups, Users, or Roles**.
2. Choose the name of the group, user, or role with the policy that you want to delete. Then choose the **Permissions** tab. If you chose **Users** or **Roles**, expand the policy.
3. To delete an inline policy in **Groups**, choose **Remove Policy**. To delete an inline policy in **Users** or **Roles**, choose **X**.

Deleting IAM Policies (AWS CLI or AWS API)

You can edit a managed or inline policy using the AWS Command Line Interface (AWS CLI) or the AWS API.

To list managed policies (AWS CLI or API)

- AWS CLI: [list-policies](#)
- AWS API: [ListPolicies](#)

To retrieve detailed information about a managed policy (AWS CLI or API)

- AWS CLI: [get-policy](#)
- AWS API: [GetPolicy](#)

To list the identities (users, groups, and roles) to which a managed policy is attached (AWS CLI or API)

- AWS CLI: [list-entities-for-policy](#)
- AWS API: [ListEntitiesForPolicy](#)

To list the managed policies attached to an identity (a user, group, or role) (AWS CLI or API)

- AWS CLI:
 - [list-attached-group-policies](#)
 - [list-attached-role-policies](#)
 - [list-attached-user-policies](#)
- AWS API:
 - [ListAttachedGroupPolicies](#)
 - [ListAttachedRolePolicies](#)
 - [ListAttachedUserPolicies](#)

To delete a customer managed policy (AWS CLI or API)

- AWS CLI: [delete-policy](#)
- AWS API: [DeletePolicy](#)

Reducing Policy Scope by Viewing User Activity

The IAM console provides information about when IAM users and roles last attempted to access AWS services. This information is called *service last accessed data*. This data can help you identify unnecessary permissions so that you can refine your IAM policies to better adhere to the principle of "least privilege." That means granting the minimum permissions required to perform a specific task.

Note

Recent activity usually appears within 4 hours. Access advisor reports activity for the last 365 days.

You can get the date and hour when an IAM entity (user, group, or role) last accessed an AWS service through IAM policy permissions. You can use this information to identify unused and not recently used permissions in your IAM policies. You can then choose to remove permissions for unused services or reorganize users with similar usage patterns into a group to help improve account security. Knowing if and when an IAM entity last exercised a permission can help you remove unnecessary permissions and tighten your IAM policies with less effort.

Important

The service last accessed data includes *all* attempts to access an AWS API, not just the successful ones. This includes all access attempts made using the AWS Management Console, the AWS API through any of the SDKs, or any of the command line tools. An unexpected entry in the service last accessed data does not mean that your account has been compromised, because the request might have been denied. Refer to your CloudTrail logs as the authoritative source for information about all API calls and whether they were successful or denied access. For more information, see [Logging IAM Events with AWS CloudTrail \(p. 262\)](#).

This topic describes the functionality of the IAM service last accessed data and how you can use it with the IAM console. It also describes two practical scenarios of using the service last accessed data to remove unnecessary permissions from an IAM policy.

Topics

- [Viewing Access Advisor Information \(p. 343\)](#)
- [Notes \(p. 344\)](#)
- [Troubleshooting Tips \(p. 344\)](#)
- [Sample Usage Scenarios \(p. 345\)](#)
- [Regions Where Data Is Tracked \(p. 346\)](#)

Viewing Access Advisor Information

You can find the data on the **Access Advisor** tab in the IAM console by examining the detail view for any IAM user, group, role, or managed policy.

Minimum permissions to see access advisor information

Users must have permission to see user, group, role, and policy details in order to view those entities in the IAM console. However, in order to view the service last accessed data, you must also have permission to use the following actions:

- `iam:GenerateServiceLastAccessedDetails`
- `iam:GetServiceLastAccessedDetails`
- `iam:GetServiceLastAccessedDetailsWithEntities`
- `iam>ListPoliciesGrantingServiceAccess`

Because these actions are not specific to a single resource, administrators should provide access to use these actions on all resources ("Resource": "*").

Be aware that granting these permissions permits the user to see which users, groups, or roles are attached to a [managed policy](#). It also allows the user to see which services a user or role has access to and which services have been accessed, by whom and when. This is similar to the `iam>ListEntitiesForPolicy` and `iam>ListAttached[User/Group/Role]Policies` permissions.

To view access advisor information

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, choose either **Groups**, or **Users**, or **Roles**, or **Policies**.
3. Choose any user, group, role, or policy name to open its detailed view and choose the **Access Advisor** tab. The tab displays a table with the following columns:

Service Name

A list of services with permissions granted by an IAM policy (if you are examining a policy), or the list of services with permissions granted to the IAM entity by all IAM policies (if you are examining a user, group, or role).

A row for a service appears if the entity is granted *any* permission to that service by an IAM policy, whether that permission is for only one action in the service or all of them.

Policies Granting Permissions

The name of one policy and a count of other policies that grant this entity permission to access the specified service. Choose the link with the policy name and count to see a list of all of

the IAM policies that grant this entity permission to access the specified service. The list also includes the name, the type of policy (AWS managed, customer managed, or inline), and the group (if applicable) through which a user gets the policy.

If you choose the name of a managed policy in the **Policies Granting Permissions** dialog box, IAM opens a new browser tab that shows the policy text. If you click a group name in the dialog box, IAM opens a new browser tab that displays the details for the group.

Last Accessed

The length of time since this user, role, or a member of a group last accessed the specified service. If the service has been updated, but it was 365 days ago or longer, this value is **365**.

Access by Members

(*Group only*) The name of one user and a count of other users who are members of the group and that have accessed this service. Choose the link to see a list of all of the IAM users who are members of the group and when each member last accessed the specified service.

If you choose the name of a user in the dialog box, IAM opens a new browser tab that displays the details for the user.

Access by Entities

(*Policy only*) The name of one user or role and a count of other users and roles who have used this policy to access the specified service. Choose the link to see a list of all of the users and roles to which this policy is attached and when each last accessed the specified service.

Additional options

- Use the **Filter** menu on the **Access Advisor** tab to limit the list to **Services accessed** and **Services not accessed**. For example, you might select **Services not accessed** for a policy to discover which services listed in that policy are never used. In that case, you might want to remove those services from the policy. You can also type a name (or part of a name) in the search box to restrict the list to only those entities whose name matches what you type.
- Choose one of the column heads to sort the information according to that column's information. Choose the header a second time to sort in the opposite order.

Notes

- The list of services in the table reflects the *current* state of your IAM policies, not any historical state. For example, if the current version of your policy allows only Amazon S3 access and it previously allowed access to other AWS services, the service last accessed data shows only a table entry for Amazon S3. If you are trying to determine the history of access control changes in your account, or want to audit historical access, we recommend that you use [AWS CloudTrail](#).
- It usually takes less than 4 hours for recent **Last Accessed** activity to appear in the table. However, under some circumstances it can take up to 12 hours. If the service is running unusually slow, a notification appears in the IAM console.
- Access advisor reports activity for the last 365 days. If there has been no activity, then Access Advisor reports **Not accessed in the tracking period**. If there has been activity, but it was 365 days ago or longer, then Access Advisor reports a last accessed period of **365**.

Troubleshooting Tips

If the service last accessed table is empty, it might be caused by one of the following:

- You selected a user that has no attached policies, either directly or through group memberships.

- You selected a managed policy attached to a group that has no members.
- You selected a user, group, or role that has neither inline nor managed policies attached.
- You selected a user that has permissions granted only by a resource-based policy.

Sample Usage Scenarios

Some examples that show the value of using the IAM access advisor for both policies (the first example) and principals (the second).

Scoping Down Permissions for an IAM Policy

Imagine an administrator who is responsible for managing a team's AWS infrastructure. The team has created an application that runs on Amazon EC2 and calls other AWS services. This means that the administrator needs to provision an EC2 instance for this application and manage its configuration. One part of this is to attach an IAM role to this EC2 instance.

Note

You can attach IAM roles to EC2 instances to enable applications that run on those instances to make API requests. That way you don't need to manually manage the security credentials that the applications use. Instead of creating and distributing your AWS credentials, you can delegate permission to make API requests to an IAM role that is assigned to the instance. For more information, see [IAM Roles for Amazon EC2](#).

However, the administrator is new to IAM and when attaching the IAM role to the EC2 instance, the administrator is not sure what to put in the role's permission policy. The administrator *wants* to implement a thorough access control mechanism, but the immediate priority is to make sure that the application works. To do that the administrator simply copies the **PowerUserAccess** AWS managed policy shown below with the intention of modifying it as it becomes clear over time what permissions are really needed. This policy grants full read-write permissions to all AWS services and resources in the account except for IAM (and is definitely *not* recommended as a long-term solution).

```
{  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "NotAction": "iam:*",  
            "Resource": "*"  
        }  
    ]  
}
```

After the application runs for a while, the administrator can use the service last accessed data to view which permissions are actually used. The administrator can then reduce the permissions for the application. The administrator signs in to the IAM console and chooses **Policies**, finds the policy attached to the IAM role associated with the EC2 instance where the application is running. The administrator then chooses the policy name to view its details and chooses the **Access Advisor** tab.

The administrator reviews the time stamps listed in the **Last Accessed** column and notices that the team's application is using only Amazon DynamoDB, Amazon S3, Amazon CloudWatch, Amazon Simple Notification Service (Amazon SNS), and Amazon Simple Queue Service (Amazon SQS). The administrator can then choose the **Permissions** tab and expand the row for a policy whose permissions need to be restricted. The administrator chooses **Edit policy** for inline policies that are attached directly to the user. To edit a managed policy, the administrator chooses the name of the policy to go to the **Policies** page. From there, administrators can revise the policy's access to include only those permissions that are required for the application to successfully run.

Scoping Down Permissions for an IAM User

Imagine an IT administrator who is responsible for ensuring that people in the organization do not have excess AWS permissions. As part of a periodic security check, the administrator reviews the permissions of all IAM users. One of these users is an application developer, who previously filled the role of a security engineer. Because of the change in job requirements, the developer is a member of both the “app-dev” group for the new job (which grants permissions to multiple services including Amazon EC2, Amazon EBS, Auto Scaling, etc.) and the “security-team” group for the old job (which grants permissions to IAM and CloudTrail).

The administrator signs into the IAM console and chooses **Users**, then chooses the name of the IAM user of the developer and then chooses the **Access Advisor** tab.

The administrator reviews the time stamps in the **Last Accessed** column and notices that the developer has not recently accessed IAM, CloudTrail, Route 53, Amazon Elastic Transcoder, and a number of other AWS services. The administrator is now ready to act on the service last accessed information. However, unlike the previous example, the next steps for a principal like the developer’s IAM user (who may be subject to multiple policies and a member of multiple IAM groups) requires the administrator to proceed with caution to avoid inadvertently disrupting other users’ access. So his first step is to determine *how* the developer is receiving these permissions.

The administrator confirms that the developer has no business need for access to IAM and CloudTrail anymore because the developer is no longer a member of the internal security team. For these permissions, after analyzing the group memberships and policies, the administrator realizes that the simplest solution is to remove the developer’s membership in the security-team IAM group, rather than make any policy changes.

The administrator might also infer that the developer’s access to Route 53, Elastic Transcoder, etc. from the app-dev group should also be revoked due to lack of use. However, before cutting these permissions out of the app-dev policy (which may adversely affect other members of the group), the administrator should consult the service last accessed data for the app-dev group policies. That data can reveal whether these permissions are universally unused by all app-dev group members or simply unused by this particular developer. If they are truly unused by all group members, then the administrator can probably safely remove those permissions from the app-dev group’s policy. If it is only this one developer that is not using the permissions, then the administrator may want to craft a different set of permissions for the developer. Another option is for the administrator to do nothing, accepting that not all users exercise all the permissions granted to them.

As this example shows, you might use the service last accessed data for principals as a starting point for a number of possible next steps, including the following suggestions. However, it’s up to you as an IAM administrator to choose the steps that strike the right balance of accessibility and least-privilege that’s appropriate to your organization.

- [Removing membership in a group \(p. 132\)](#)
- [Detaching a managed policy \(p. 332\)](#)
- [Deleting a managed policy \(p. 341\)](#)
- [Deleting an inline policy and converting to a managed policy \(p. 341\)](#)
- [Editing an existing policy to remove permissions \(p. 337\)](#)
- [Adding an explicit deny to an existing policy \(p. 462\)](#)

Regions Where Data Is Tracked

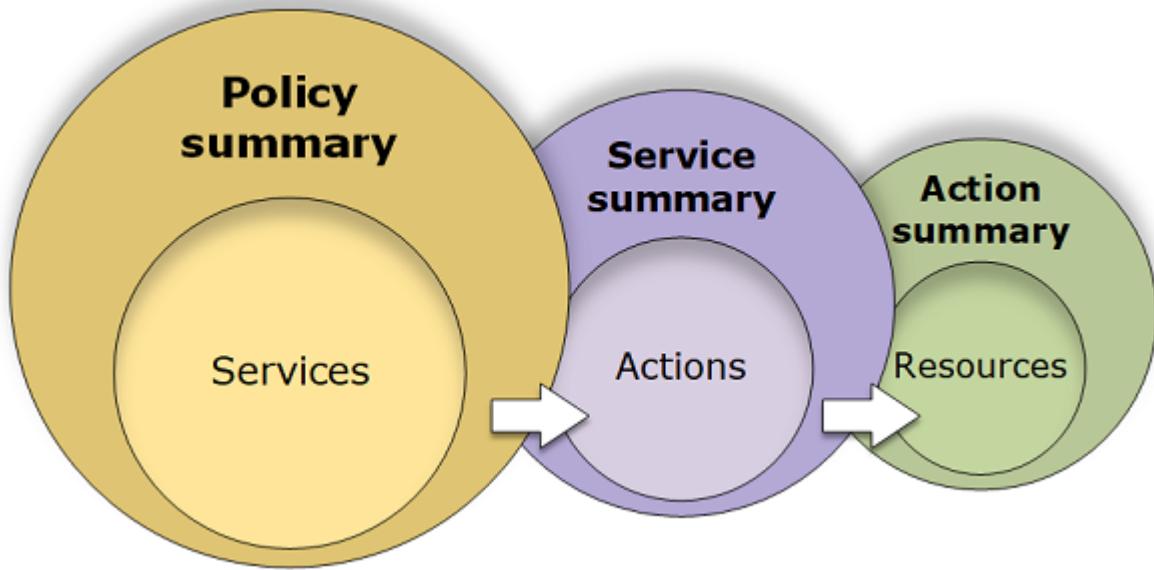
AWS collects service last accessed data in most regions. As AWS adds support for service last accessed to additional regions, those regions are added to this list:

Region name	Region
US East (N. Virginia)	us-east-1
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
Asia Pacific (Tokyo)	ap-northeast-1
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
EU (Frankfurt)	eu-central-1
EU (Ireland)	eu-west-1
South America (São Paulo)	sa-east-1
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Mumbai)	ap-south-1

If a region is not listed in the previous table, then that region does not yet provide service last accessed data.

Understanding Permissions Granted by a Policy

The IAM console includes *policy summary* tables that describe the access level, resources, and conditions that are allowed or denied for each service in a policy. Policies are summarized in three tables: the [policy summary \(p. 348\)](#), the [service summary \(p. 358\)](#), and the [action summary \(p. 362\)](#). The *policy summary* table includes a list of services. Choose a service there to see the *service summary*. This summary table includes a list of the actions and associated permissions for the chosen service. You can choose an action from that table to view the *action summary*. This table includes a list of resources and conditions for the chosen action.



You can view policy summaries on the **Users** page or **Roles** page for all policies (managed and inline) that are attached to that user. View summaries on the **Policies** page for all managed policies. Managed policies include AWS managed policies, AWS managed job function policies, and customer managed policies. You can view summaries for these policies on the **Policies** page regardless of whether they are attached to a user or other IAM identity.

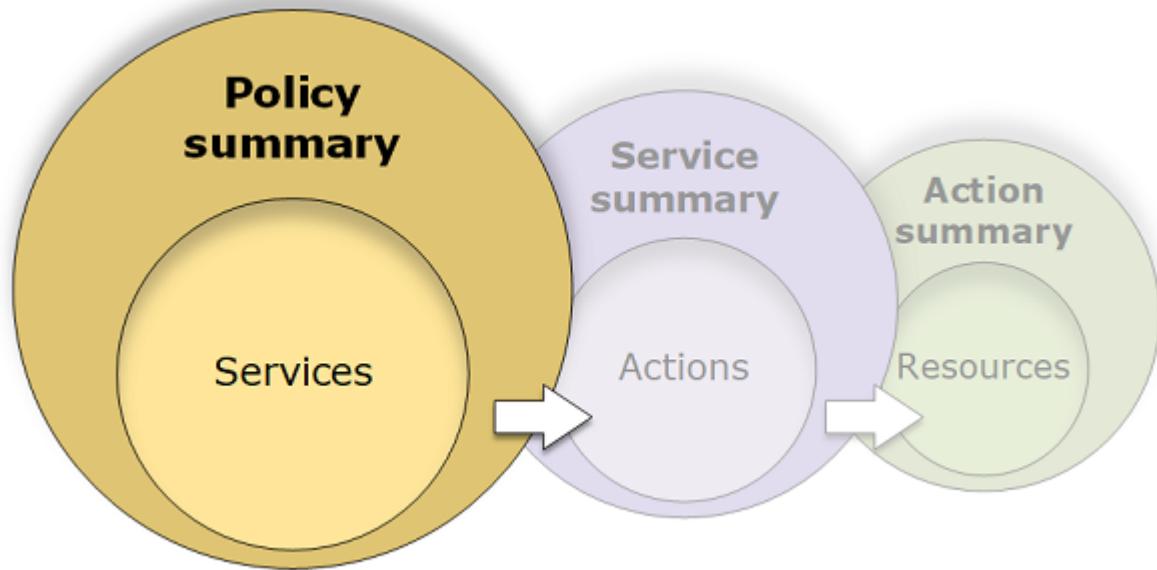
You can use the information in the policy summaries to understand the permissions that are allowed or denied by your policy. Policy summaries can help you [troubleshoot \(p. 386\)](#) and fix policies that are not providing the permissions that you expect.

Topics

- [Policy Summary \(List of Services\) \(p. 348\)](#)
- [Service Summary \(List of Actions\) \(p. 358\)](#)
- [Action Summary \(List of Resources\) \(p. 362\)](#)
- [Examples of Policy Summaries \(p. 365\)](#)

Policy Summary (List of Services)

Policies are summarized in three tables: the policy summary, the [service summary \(p. 358\)](#), and the [action summary \(p. 362\)](#). The *policy summary* table includes a list of services and summaries of the permissions that are defined by the chosen policy.



The policy summary table is grouped into one or more **Uncategorized services**, **Explicit deny**, and **Allow** sections. If the policy includes a service that IAM does not recognize, then the service is included in the **Uncategorized services** section of the table. If IAM recognizes the service, then it is included under the **Explicit deny** or **Allow** sections of the table, depending on the effect of the policy (Deny or Allow).

Viewing Policy Summaries

You can view the summaries for any policies that are attached to a user on the **Users** page. You can view the summaries for any policies that are attached to a role on the **Roles** page. You can view the policy summary for managed policies on the **Policies** page. If your policy does not include a policy summary, see [Missing Policy Summary \(p. 390\)](#) to learn why.

To view the policy summary from the Policies page

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, choose **Policies**.
3. In the list of policies, choose the name of the policy that you want to view.
4. On the **Summary** page for the policy, view the **Permissions** tab to see the policy summary.

To view the summary for a policy attached to a user

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Choose **Users** from the navigation pane.
3. In the list of users, choose the name of the user whose policy you want to view.
4. On the **Summary** page for the user, view the **Permissions** tab to see the list of policies that are attached to the user directly or from a group.
5. In the table of policies for the user, expand the row of the policy that you want to view.

To view the summary for a policy attached to a role

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, choose **Roles**.
3. In the list of roles, choose the name of the role whose policy you want to view.
4. On the **Summary** page for the role, view the **Permissions** tab to see the list of policies that are attached to the role.
5. In the table of policies for the role, expand the row of the policy that you want to view.

Editing Policies to Fix Warnings

While viewing a policy summary, you might find a typo or notice that the policy does not provide the permissions that you expected. You cannot edit a policy summary directly. However, you can edit a managed policy using the visual policy editor, which catches many of the same errors and warnings that the policy summary reports. You can then view the changes in the policy summary to confirm that you fixed all of the issues. To learn how to edit an inline policy, see [the section called "Editing IAM Policies" \(p. 337\)](#). You cannot edit AWS managed policies.

To edit a policy for your policy summary using the Visual editor tab

1. Open the policy summary as explained in the previous procedures.
2. Choose **Edit policy**.

If you are on the **Users** page and choose to edit a customer managed policy that is attached to that user, you are redirected to the **Policies** page. You can edit customer managed policies only on the **Policies** page.

3. Choose the **Visual editor** tab to view the editable visual representation of your policy. IAM might restructure your policy to optimize it for the visual editor and to make it easier for you to find and fix any problems. The warnings and error messages on the page can guide you to fix any issues with your policy. For more information about how IAM restructures policies, see [Policy Restructuring \(p. 387\)](#).
4. Edit your policy and choose **Review policy** to see your changes reflected in the policy summary. If you still see a problem, choose **Previous** to return to the editing screen.
5. Choose **Save** to save your changes.

To edit a policy for your policy summary using the JSON tab

1. Open the policy summary as explained in the previous procedures.
2. Choose **{ } JSON** and **Policy summary** to compare the policy summary to the JSON policy document. You can use this information to determine which lines in the policy document you want to change.
3. Choose **Edit policy** and then choose the **JSON** tab to edit the JSON policy document.

Note

You can switch between the **Visual editor** and **JSON** tabs any time. However, if you make changes or choose **Review policy** in the **Visual editor** tab, IAM might restructure your policy to optimize it for the visual editor. For more information, see [Policy Restructuring \(p. 387\)](#).

If you are on the **Users** page and choose to edit a customer managed policy that is attached to that user, you are redirected to the **Policies** page. You can edit customer managed policies only on the **Policies** page.

4. Edit your policy and choose **Review policy** to see your changes reflected in the policy summary. If you still see a problem, choose **Previous** to return to the editing screen.

5. Choose **Save** to save your changes.

Understanding the Elements of a Policy Summary

In the following example of a user details page, the **PolSumUser** user has eight attached policies. The **SummaryAllElements** policy is a managed policy (customer managed policy) that is attached directly to the user. This policy is expanded to show the policy summary. To view the JSON policy document for this policy, see [the section called “SummaryAllElements JSON Policy Document” \(p. 355\)](#).

The screenshot shows the AWS IAM User Details page for the user **PolSumUser**. The top navigation bar includes tabs for **Permissions**, **Groups (1)**, **Security credentials**, and **Access Advisor**. The **Permissions** tab is selected, indicated by a red circle labeled **1**. Below this, there is a button to **Add permissions** and a message stating **Attached policies: 8**. A table lists the attached policies, with the **SummaryAllElements** policy highlighted by a red circle labeled **2** and identified as a **Managed policy**. A warning message in an orange box states: **This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition. For details, choose Show remaining. Learn more**. Below the table, there is a **Policy summary** button with a red circle labeled **4**, a **{ } JSON** button, an **Edit policy** button, a **Simulate policy** button with a red circle labeled **5**, and a **Filter** input field with a red circle labeled **6**. A large red box labeled **7** highlights the **Policy summary** table, which contains the following data:

Service	Access level	Resource	Request condition
<u>Unrecognized services</u>			
codeddeploy	⚠		
Explicit deny (1 of 103 services)			
S3	⚠	Full: Read, Write, Permissions management Limited: List	Multiple
Allow (3 of 103 services) Show remaining 100			
Billing	Full: Read Limited: Write	All resources	Multiple
EC2	⚠	None	All resources
S3	⚠	Limited: Write, Permissions management	BucketName = developer_bucket, ObjectPath = All s3:x-amz-acl = public-read

At the bottom of the table, there are links for **EC2 Troubleshoot** and **Managed policy**.

In the preceding image, the policy summary is visible from within the user details page:

1. The **Permissions** tab for a user includes the policies that are attached to the **PolSumUser** user.
2. The **SummaryAllElements** policy is one of several policies that are attached to the user. The policy is expanded in order to view the policy summary.
3. If the policy does not grant permissions to all the actions, resources, and conditions defined in the policy, then a warning or error banner appears at the top of the page. The policy summary then includes details about the problem. To learn how policy summaries help you to understand and troubleshoot the permissions that your policy grants, see [the section called “My Policy Does Not Grant the Expected Permissions” \(p. 392\)](#).

4. Use the **Policy summary** and **{ } JSON** buttons to toggle between the policy summary and the JSON policy document.
5. **Simulate policy** opens the policy simulator for testing the policy.
6. Use the search box to reduce the list of services and easily find a specific service.
7. The expanded view shows additional details of the **SummaryAllElements** policy.

The following policy summary table image shows the expanded **SummaryAllElements** policy on the **PolSumUser** user details page.

A Service	G Access level	H Resource	I Request condition
B Unrecognized services			
codeddeploy ⚠			
C Explicit deny (1 of 103 services)			
S3 ⚠	Full: Read, Write, Permissions management Limited: List	Multiple	None
D Allow (3 of 103 services) Show remaining 100			
Billing	Full: Read Limited: Write	All resources	Multiple
E EC2 ⚠	None	All resources	None
F S3 ⚠	Limited: Write, Permissions management	BucketName = developer_bucket, ObjectPath = All	s3:x-amz-acl = public-read

In the preceding image, the policy summary is visible from within the user details page:

- A. **Service** – This column lists the services that are defined within the policy and provides details for each service. Each service name in the policy summary table is a link to the *service summary* table, which is explained in [Service Summary \(List of Actions\) \(p. 358\)](#). In this example, permissions are defined for the Amazon S3, Billing, and Amazon EC2 services. The policy also defines permissions for a (misspelled) codedploy service, which IAM does not recognize.
- B. **Unrecognized services** – This policy includes an unrecognized service (in this case **codedploy** ⚠). You can use this warning to check whether a service name might include a typo. If the service name is correct, then the service might not support policy summaries, might be in preview, or might be a custom service. To request policy summary support for a generally available (GA) service, see [Service Does Not Support IAM Policy Summaries \(p. 391\)](#). In this example, the policy includes an unrecognized codedploy service that is missing an e. Because of this typo, the policy does not provide the expected AWS CodeDeploy permissions. You can [edit the policy \(p. 350\)](#) to include the accurate codedploy service name; the service then appears in the policy summary.
- C. For those services that IAM recognizes, it arranges services according to whether the policy allows or explicitly denies the use of the service. In this example, the policy includes Allow and Deny statements for the Amazon S3 service. Therefore the policy summary includes S3 within both the **Explicit deny** and **Allow** sections.
- D. **Show remaining 100** – Choose this link to expand the table to include the services that are not defined by the policy. These services are *implicitly denied* (or denied by default) within this policy. However, a statement in another policy might still allow or explicitly deny using the service. The policy summary summarizes the permissions of a single policy. To learn about how the AWS service decides whether a given request should be allowed or denied, see [IAM JSON Policy Evaluation Logic \(p. 458\)](#).
- E. **EC2 ⚠** – This service includes an unrecognized action. IAM recognizes service names, actions, and resource types for services that support policy summaries. When a service is recognized but contains an action that is not recognized, IAM includes a warning next to that service. In this example, IAM can't recognize at least one Amazon EC2 action. To learn more about unrecognized actions and to view the unrecognized action in an S3 service summary, see [Service Summary \(List of Actions\) \(p. 358\)](#).

Note

IAM reviews service names, actions, and resource types for services that support policy summaries. However, your policy summary might include a resource value or condition that does not exist. Always test your policies with the [policy simulator \(p. 322\)](#).

F.



S3 – This service includes an unrecognized resource. IAM recognizes service names, actions, and resource types for services that support policy summaries. When a service is recognized but contains a resource type that is not recognized, IAM includes a warning next to that service. In this example, IAM can't recognize at least one Amazon S3 action. To learn more about unrecognized resources and to view the unrecognized resource type in an S3 service summary, see [Service Summary \(List of Actions\) \(p. 358\)](#).

G. **Access level** – This column tells whether the actions in each access level (List, Read, Write, and Permissions management) have Full or Limited permissions defined in the policy. For additional details and examples of the access level summary, see [Understanding Access Level Summaries Within Policy Summaries \(p. 357\)](#).

- **Full access** – This entry indicates that the service has access to all actions within all four of the access levels available for the service. In this example, because this row is in the **Explicit deny** section of the table, all Amazon S3 actions are denied for the resources included in the policy.
- If the entry does not include **Full access**, then the service has access to some but not all of the actions for the service. The access is then defined by following descriptions for each of the four access level classifications (List, Read, Write, and Permissions management):

Full: The policy provides access to all actions within each access level classification listed. In this example, the policy provides access to all of the Billing Read actions.

Limited: The policy provides access to one or more but not all actions within each access level classification listed. In this example, the policy provides access to some of the Billing Write actions.

H. **Resource** – This column shows the resources that the policy specifies for each service.

- **Multiple** – The policy includes more than one but not all of the resources within the service. In this example, access is explicitly denied to more than one Amazon S3 resource.
- **All resources** – The policy is defined for all resources within the service. In this example, the policy allows the listed actions to be performed on all Billing resources.
- Resource text – The policy includes one resource within the service. In this example, the listed actions are allowed on only the developer_bucket Amazon S3 bucket resource. Depending on the information that the service provides to IAM, you might see an ARN such as `arn:aws:s3:::developer_bucket/*`, or you might see the defined resource type, such as `BucketName = developer_bucket`.

Note

This column can include a resource from a different service. If the policy statement that includes the resource does not include both actions and resources from the same service, then your policy includes mismatched resources. IAM does not warn you about mismatched resources when you create a policy, or when you view a policy in the policy summary. If this column includes a mismatched resource, then you should review your policy for errors. To better understand your policies, always test them with the [policy simulator \(p. 322\)](#).

I. **Request condition** – This column indicates whether the services or actions associated with the resource are subject to conditions.

- **None** – The policy includes no conditions for the service. In this example no conditions are applied to the denied actions in the Amazon S3 service.
- Condition text – The policy includes one condition for the service. In this example, the listed **Billing** actions are allowed only if the IP address of the source matches `203.0.113.0/24`.
- **Multiple** – The policy includes more than one condition for the service. In this example, access to the listed Amazon S3 actions is allowed based on more than one condition. To view each of the multiple conditions for the policy, choose `{ } JSON` to view the policy document.

When a policy or an element within the policy does not grant permissions, IAM provides additional warnings and information in the policy summary. The following policy summary table shows the expanded **Show remaining 100 services** on the **PolSumUser** user details page with the possible warnings.

Service	Access level	a Resource	b Request condition
<u>Unrecognized services</u>			
codeddeploy ⚠			
<u>Explicit deny (1 of 103 services)</u>			
S3 ⚠	Full: Read, Write, Permissions management Limited: List	Multiple ⚠ One or more actions do not have an applicable resource.	None
<u>Allow (3 of 103 services) Hide remaining 100</u>			
d ... e Billing	Full: Read Limited: Write	All resources	Multiple
CodeBuild	f ⚠ None - No actions are defined.	arn:aws:codebuild:us-east-1:123456789012:project/my-demo-project	None
CodeCommit	None	g ⚠ No resources are defined.	None
CodeDeploy	⚠ None - No actions are defined.	arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup:*	None
EC2 ⚠	None	All resources	None
S3 ⚠	Limited: Write, Permissions management	BucketName = developer_bucket, ObjectPath = All ⚠ One or more resources do not have an applicable action.	s3:x-amz-acl = public-read h ⚠ One or more conditions do not have an applicable action.

In the preceding image, you can see all services that include defined actions, resources, or conditions with no permissions:

- a. **Resource warnings** – For services that do not provide permissions for all of the included actions or resources, you see one of the following warnings in the **Resource** column of the table:

- ⚠ **No resources are defined.** – This means that the service has defined actions but no supported resources are included in the policy.
- ⚠ **One or more actions do not have an applicable resource.** – This means that the service has defined actions, but that some of those actions don't have a supported resource.
- ⚠ **One or more resources do not have an applicable action.** – This means that the service has defined resources, but that some of those resources don't have a supporting action.

If a service includes both actions that do not have an applicable resource and resources that do not have an applicable resource, then only the **One or more resources do not have an applicable action** warning is shown. This is because when you view the service summary for the service, resources that do not apply to any action are not shown. For the `ListAllMyBuckets` action, this policy includes the last warning because the action does not support resource-level permissions, and does not support the `s3:x-amz-acl` condition key. If you fix either the resource problem or the condition problem, the remaining issue appears in a detailed warning.

- b. **Request condition warnings** – For services that do not provide permissions for all of the included conditions, you see one of the following warnings in the **Request condition** column of the table:

- ⚠ **One or more actions do not have an applicable condition.** – This means that the service has defined actions, but that some of those actions don't have a supported condition.

-  **One or more conditions do not have an applicable action.** – This means that the service has defined conditions, but that some of those conditions don't have a supporting action,
- c. **Multiple |  One or more actions do not have an applicable resource.** – The Deny statement for Amazon S3 includes more than one resource. It also includes more than one action, and some actions support the resources and some do not. To view this policy, see [the section called "SummaryAllElements JSON Policy Document" \(p. 355\)](#). In this case, the policy includes all Amazon S3 actions, and only the actions that can be performed on a bucket or bucket object are denied.
 - d. The ellipses (...) indicate that all the services are included in the page, but we are showing only the rows with information relevant to this policy. When you view this page in the AWS Management Console, you see all the AWS services.
 - e. The background color in the table rows indicates services that do not grant any permissions. You cannot get any additional information about these services in the policy summary. For services in white rows, you can choose the name of the service to view the service summary (list of actions) page. There you can learn more about the permissions granted for that service.
 - f.  **None - No actions are defined.** – This means that the service is defined as a resource or condition, but that no actions are included for the service, and therefore the service provides no permissions. In this case, the policy includes a AWS CodeBuild resource but no AWS CodeBuild actions.
 - g.  **No resources are defined** – The service has defined actions, but no supported resources are included in the policy, and therefore the service provides no permissions. In this case, the policy includes AWS CodeCommit actions but no AWS CodeCommit resources.
 - h. **BucketName = developer_bucket, ObjectPath = All |  One or more resources do not have an applicable action.** – The service has a defined bucket object resource, and at least one more resource that does not have a supporting action.
 - i. **s3:x-amz-acl = public-read |  One or more conditions do not have an applicable action.** – The service has a defined s3:x-amz-acl condition key, and at least one more condition key that does not have a supporting action.

SummaryAllElements JSON Policy Document

The **SummaryAllElements** policy is not intended for you to use to define permissions in your account. Rather, it is included to demonstrate the errors and warnings that you might encounter while viewing a policy summary.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "aws-portal:ViewBilling",
                "aws-portal:ViewPaymentMethods",
                "aws-portal:ModifyPaymentMethods",
                "aws-portal:ViewAccount",
                "aws-portal:ModifyAccount",
                "aws-portal:ViewUsage"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "IpAddress": {
                    "NotIpAddress": [
                        "127.0.0.1/32"
                    ]
                }
            }
        }
    ]
}
```

```
        "aws:SourceIp": "203.0.113.0/24"
    }
}
{
    "Effect": "Deny",
    "Action": [
        "s3:*"
    ],
    "Resource": [
        "arn:aws:s3::::customer",
        "arn:aws:s3::::customer/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:GetConsoleScreenshots"
    ],
    "Resource": [
        "*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "codeddeploy:*",
        "codecommit:*"
    ],
    "Resource": [
        "arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup:*",
        "arn:aws:codebuild:us-east-1:123456789012:project/my-demo-project"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "s3>ListAllMyBuckets",
        "s3GetObject",
        "s3DeleteObject",
        "s3PutObject",
        "s3PutObjectAcl"
    ],
    "Resource": [
        "arn:aws:s3::::developer_bucket",
        "arn:aws:s3::::developer_bucket/*",
        "arn:aws:autoscaling:us-east-2:123456789012:autoscalgrp"
    ],
    "Condition": {
        "StringEquals": {
            "s3:x-amz-acl": [
                "public-read"
            ],
            "s3:prefix": [
                "custom",
                "other"
            ]
        }
    }
}
]
```

Understanding Access Level Summaries Within Policy Summaries

Policy summaries include an access level summary that describes the action permissions defined for each service that is mentioned in the policy. To learn about policy summaries, see [Understanding Permissions Granted by a Policy \(p. 347\)](#). Access level summaries indicate whether the actions in each access level (List, Read, Write, and Permissions management) have Full or Limited permissions defined in the policy. To view a list of actions that belong to each of the action levels for a specific service, see [IAM Policy Actions Grouped by Access Level \(p. 615\)](#). To see a complete list of actions for a specific service, see [AWS Service Actions and Condition Context Keys for Use in IAM Policies \(p. 485\)](#).

The following example describes the access provided by a policy for the given services. For examples of full JSON policy documents and their related summaries, see [Examples of Policy Summaries \(p. 365\)](#).

Service	Access level	This policy provides:
IAM	Full access	Access to all actions within the IAM service
CloudWatch	Full: List	Access to all CloudWatch actions in the List access level, but no access to actions with the Read, Write, or Permissions management access level classification
Data Pipeline	Limited: List, Read	Access to at least one but not all AWS Data Pipeline actions in the List and Read access level, but not the Write or Permissions management actions
EC2	Full: List, Read Limited: Write	Access to all Amazon EC2 List and Read actions and access to at least one but not all Amazon EC2 Write actions, but no access to actions with the Permissions management access level classification
S3	Limited: Read, Write, Permissions management	Access to at least one but not all Amazon S3 Read, Write and Permissions management actions
codedploy	(empty)	Unknown access, because IAM does not recognize this service
API Gateway	None	No access is defined in the policy
CodeBuild	 No actions are defined.	No access because no actions are defined for the service. To learn how to understand and troubleshoot this issue, see the section called "My Policy Does Not Grant the Expected Permissions" (p. 392)

As [previously mentioned \(p. 353\)](#), **Full access** indicates that the policy provides access to all the actions within the service. Policies that provide access to some but not all actions within a service are further grouped according to the access level classification. This is indicated by one of the following access-level groupings:

- **Full:** The policy provides access to all actions within the specified access level classification.

- **Limited**: The policy provides access to one or more but not all actions within the specified access level classification.
- **None**: The policy provides no access.
- **(empty)**: IAM does not recognize this service. If the service name includes a typo, then the policy provides no access to the service. If the service name is correct, then the service might not support policy summaries or might be in preview. In this case, the policy might provide access, but that access cannot be shown in the policy summary. To request policy summary support for a generally available (GA) service, see [Service Does Not Support IAM Policy Summaries \(p. 391\)](#).

Access level summaries that include partial access to actions are grouped using the following access level classifications:

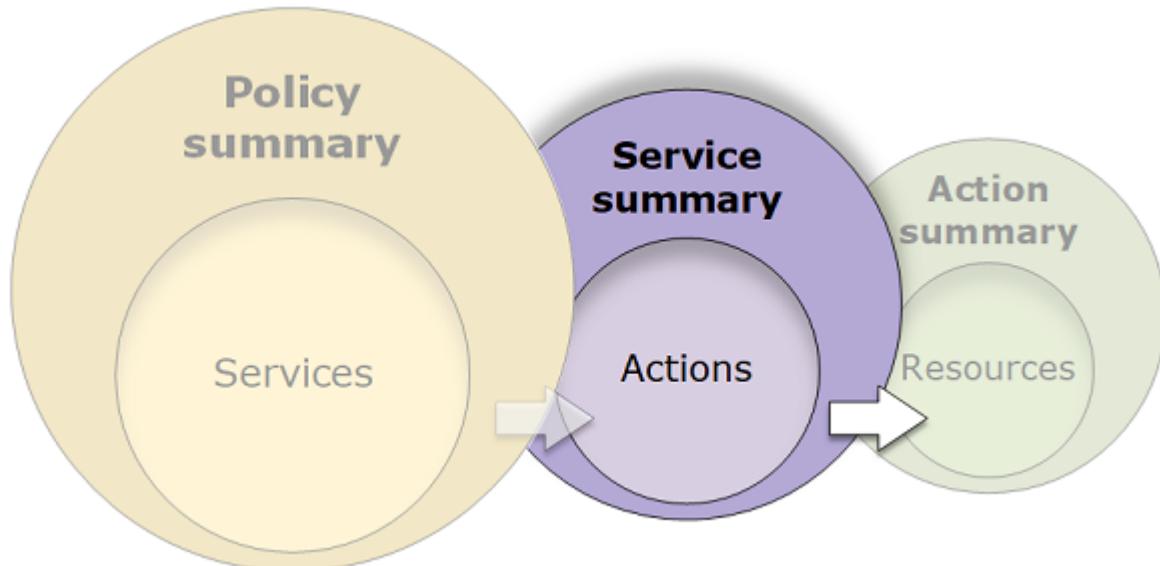
- **List (p. 616)**: Permission to list resources within the service to determine whether an object exists. Actions with this level of access can list objects but cannot see the contents of a resource. For example, the Amazon S3 action `ListBucket` has the **List** access level.
- **Read (p. 637)**: Permission to read but not edit the contents and attributes of resources in the service. For example, the Amazon S3 actions `GetObject` and `GetBucketLocation` have the **Read** access level.
- **Write (p. 663)**: Permission to create, delete, or modify resources in the service. For example, the Amazon S3 actions `CreateBucket`, `DeleteBucket` and `PutObject` have the **Write** access level.
- **Permissions management (p. 710)**: Permission to grant or modify resource permissions in the service. For example, most IAM and AWS Organizations actions, as well as actions like the Amazon S3 actions `PutBucketPolicy` and `DeleteBucketPolicy` have the **Permissions management** access level.

Tip

To improve the security of your AWS account, restrict or regularly monitor policies that include the **Permissions management** access level classification.

Service Summary (List of Actions)

Policies are summarized in three tables: the [policy summary \(p. 348\)](#), the service summary, and the [action summary \(p. 362\)](#). The *service summary* table includes a list of the actions and summaries of the permissions that are defined by the policy for the chosen service.



You can view a service summary for each service listed in the policy summary that grants permissions. The table is grouped into **Uncategorized actions**, **Uncategorized resource types**, and access level sections. If the policy includes an action that IAM does not recognize, then the action is included in the **Uncategorized actions** section of the table. If IAM recognizes the action, then it is included under one of the access level (**List**, **Read**, **Write** and **Permissions management**) sections of the table. To view a list of actions that belong to each of the action levels for a specific service, see [IAM Policy Actions Grouped by Access Level \(p. 615\)](#). To see a complete list of actions for a specific service, see [AWS Service Actions and Condition Context Keys for Use in IAM Policies \(p. 485\)](#).

Viewing Service Summaries

You can view the service summary for managed policies on the **Policies** page, or view service summaries for inline and managed policies attached to a user or role through the **Users** page and **Roles** page. However, if you choose a service name on the **Users** page or **Roles** page from a managed policy, you are redirected to the **Policies** page. Service summaries for managed policies must be viewed on the **Policies** page.

To view the service summary for a managed 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, choose **Policies**.
3. In the list of policies, choose the name of the policy that you want to view.
4. On the **Summary** page for the policy, view the **Permissions** tab to see the policy summary.
5. In the policy summary list of services, choose the name of the service that you want to view.

To view the service summary for a policy attached to a user

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, choose **Users**.
3. In the list of users, choose the name of the user whose policy you want to view.
4. On the **Summary** page for the user, view the **Permissions** tab to see the list of policies that are attached to the user directly or from a group.
5. In the table of policies for the user, expand the row of the policy that you want to view.
6. In the policy summary list of services, choose the name of the service that you want to view.

Note

If the policy that you select is an inline policy that is attached directly to the user, then the service summary table appears. If the policy is an inline policy attached from a group, then you are taken to the JSON policy document for that group. If the policy is a managed policy, then you are taken to the service summary for that policy on the **Policies** page.

To view the service summary for a policy attached to a role

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Choose **Roles** from the navigation pane.
3. In the list of roles, choose the name of the role whose policy you want to view.
4. On the **Summary** page for the role, view the **Permissions** tab to see the list of policies that are attached to the role.
5. In the table of policies for the role, expand the row of the policy that you want to view.
6. In the policy summary list of services, choose the name of the service that you want to view.

Understanding the Elements of a Service Summary

The example below is the service summary for Amazon S3 actions that are allowed from the **SummaryAllElements** policy summary (see [the section called “SummaryAllElements JSON Policy Document” \(p. 355\)](#)). The actions for this service are grouped by **Uncategorized actions**, **Uncategorized resource types**, and access level. For example, two **Write** actions are defined out of the total 29 **Write** actions available for the service.

Action	Resource	Condition
Unrecognized actions		
DeleteObject		
List (0 of 4 actions)		
... (No access)		
ListAllMyBuckets (No access)		This action does not have an applicable resource and condition.
Read (0 of 30 actions)		
GetObject (No access)	BucketName = developer_bucket, ObjectPath = All	s3:x-amz-acl = public-read is not a supported condition key for this action.
Write (1 of 29 actions)		
PutObject	BucketName = developer_bucket, ObjectPath = All	s3:x-amz-acl = public-read
Permissions management (1 of 6 actions)		
PutObjectAcl	BucketName = developer_bucket, ObjectPath = All	s3:x-amz-acl = public-read

The service summary page for a managed policy includes the following information:

- If the policy does not grant permissions to all the actions, resources, and conditions defined for the service in the policy, then a warning banner appears at the top of the page. The service summary then includes details about the problem. To learn how policy summaries help you to understand and troubleshoot the permissions that your policy grants, see [the section called “My Policy Does Not Grant the Expected Permissions” \(p. 392\)](#).
- Next to the **Back** link appears the name of the service (in this case S3). The service summary for this service includes the list of allowed actions that are defined in the policy. If instead, the text **(Explicitly denied)** appears next to the name of a service, then the actions listed in the service summary table are explicitly denied.
- Choose **{ } JSON** to see additional details about the policy. You can do this to view all conditions that are applied to the actions. (If you are viewing the service summary for an inline policy that is attached directly to a user, you must close the service summary dialog box and return to the policy summary to access the JSON policy document.)
- To view the summary for a specific action, type keywords into the search box to reduce the list of available actions.
- Action (2 of 69 actions)** – This column lists the actions that are defined within the policy and provides the resources and conditions for each action. If the policy grants permissions to the action, then

the action name links to the [action summary \(p. 362\)](#) table. The count indicates the number of recognized actions that provide permissions. The total is the number of known actions for the service. In this example, 2 actions provide permissions out of 69 total known S3 actions.

6. **Show/Hide remaining 67** – Choose this link to expand or hide the table to include actions that are known but do not provide permissions for this service. Expanding the link also displays warnings for any elements that do not provide permissions.
7. **Unrecognized resource types** – This policy includes at least one unrecognized resource type within the policy for this service. You can use this warning to check whether a resource type might include a typo. If the resource type is correct, then the service might not fully support policy summaries, might be in preview, or might be a custom service. To request policy summary support for a specific resource type in a generally available (GA) service, see [Service Does Not Support IAM Policy Summaries \(p. 391\)](#). In this example, the autoscaling service name is missing an a.
8. **Unrecognized actions** – This policy includes at least one unrecognized action within the policy for this service. You can use this warning to check whether an action might include a typo. If the action name is correct, then the service might not fully support policy summaries, might be in preview, or might be a custom service. To request policy summary support for a specific action in a generally available (GA) service, see [Service Does Not Support IAM Policy Summaries \(p. 391\)](#). In this example,

 the DeleteObject action is missing an e.

Note

IAM reviews service names, actions, and resource types for services that support policy summaries. However, your policy summary might include a resource value or condition that does not exist. Always test your policies with the [policy simulator \(p. 322\)](#).

9. For those actions that IAM recognizes, the table groups these actions into at least one or up to four sections, depending on the level of access that the policy allows or denies. The sections are **List**, **Read**, **Write**, and **Permissions management**. You can also see the number of actions that are defined out of the total number of actions available within each access level. For information about which actions belong to each of the action levels for AWS services, see [IAM Policy Actions Grouped by Access Level \(p. 615\)](#). To see a complete list of actions for a specific service, see [AWS Service Actions and Condition Context Keys for Use in IAM Policies \(p. 485\)](#).

10. The ellipses (...) indicate that all the actions are included in the page, but we are showing only the rows with information relevant to this policy. When you view this page in the AWS Management Console, you see all the actions for your service.

11. **(No access)** – This policy includes an action that does not provide permissions.

12. Actions that provide permissions include a link to the action summary.

13. **Resource** – This column shows the resources that the policy defines for the service. IAM does not check whether the resource applies to each action. In this example, actions in the S3 service are allowed on only the developer_bucket Amazon S3 bucket resource. Depending on the information that the service provides to IAM, you might see an ARN such as arn:aws:s3:::developer_bucket/*, or you might see the defined resource type, such as BucketName = developer_bucket.

Note

This column can include a resource from a different service. If the policy statement that includes the resource does not include both actions and resources from the same service, then your policy includes mismatched resources. IAM does not warn you about mismatched resources when you create a policy, or when you view a policy in the service summary. IAM also does not indicate whether the action applies to the resources, only whether the service matches. If this column includes a mismatched resource, then you should review your policy for errors. To better understand your policies, always test them with the [policy simulator \(p. 322\)](#).

14. **Resource warning** – For actions with resources that do not provide full permissions, you see one of the following warnings:

- **This action does not support resource-level permissions. This requires a wildcard (*) for the resource.** – This means that the policy includes resource-level permissions but must include "Resource": ["*"] to provide permissions for this action.
- **This action does not have an applicable resource.** – This means that the action is included in the policy without a supported resource.
- **This action does not have an applicable resource and condition.** – This means that the action is included in the policy without a supported resource and without a supported condition. In this case, there is also condition included in the policy for this service, but there are no conditions that apply to this action.

For the `ListAllMyBuckets` action, this policy includes the last warning because the action does not support resource-level permissions and does not support the `s3:x-amz-acl` condition key. If you fix either the resource problem or the condition problem, the remaining issue appears in a detailed warning.

15**Request condition** – This column tells whether the actions associated with the resource are subject to conditions. To learn more about those conditions, choose `{ } JSON` to review the JSON policy document.

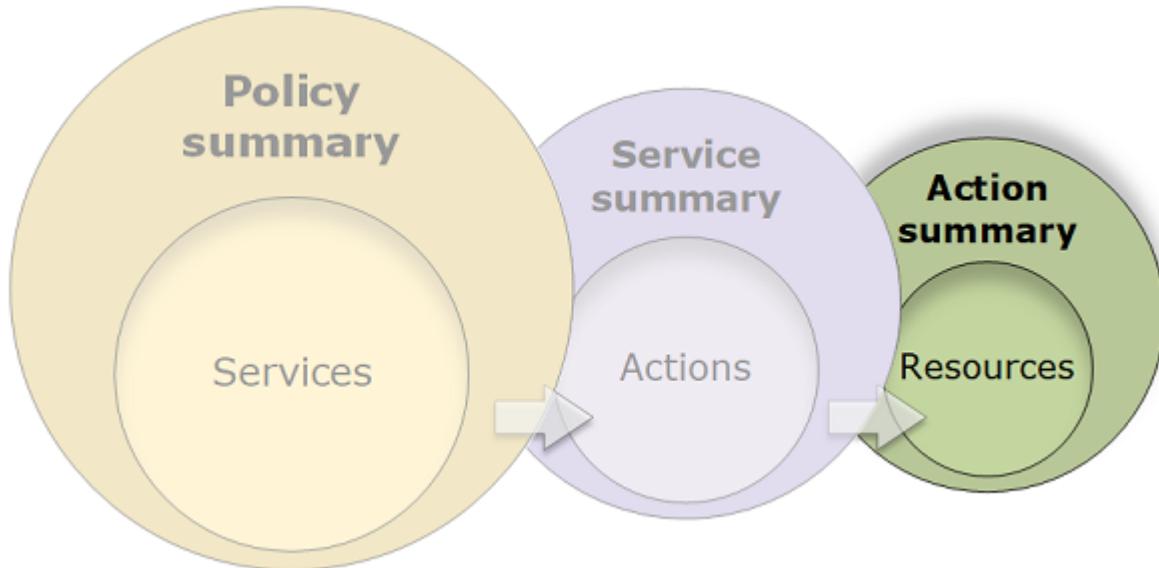
16**Condition warning** – For actions with conditions that do not provide full permissions, you see one of the following warnings:

- **<CONDITION_KEY> is not a supported condition key for this action.** – This means that the policy includes a condition key for the service that is not supported for this action.
- **Multiple condition keys are not supported for this action.** – This means that the policy includes more than one condition keys for the service that are not supported for this action.

For `GetObject`, this policy includes the `s3:x-amz-acl` condition key, which will not work with this action. Although the action supports the resource, the policy does not grant any permissions for this action because the condition will never be true for this action.

Action Summary (List of Resources)

Policies are summarized in three tables: the [policy summary \(p. 348\)](#), the [service summary \(p. 358\)](#), and the [action summary](#). The *action summary* table includes a list of resources and the associated conditions that apply to the chosen action.



To view an action summary for each action that grants permissions, choose the link in the service summary. The action summary table includes details about the resource, including its **Region** and **Account**. You can also view the conditions that apply to each resource. This shows you conditions that apply to some resources but not others.

Viewing Action Summaries

You can view the action summary for any policy that is attached to a user on the **Users** page. You can view the action summary for any policy that is attached to a role on the **Roles** page. You can view the action summary for managed policies on the **Policies** page. However, if you try to view the action summary for a managed policy from the **Users** page or the **Roles** page, you are redirected to the **Policies** page.

To view the action summary for a managed 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, choose **Policies**.
3. In the list of policies, choose the name of the policy that you want to view.
4. On the **Summary** page for the policy, view the **Permissions** tab to see the policy summary.
5. In the policy summary list of services, choose the name of the service that you want to view.
6. In the service summary list of actions, choose the name of the action that you want to view.

To view the action summary for a policy attached to a user

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Choose **Users** from the navigation pane.
3. In the list of users, choose the name of the user whose policy you want to view.
4. On the **Summary** page for the user, view the **Permissions** tab to see the list of policies that are attached to the user directly or from a group.
5. In the table of policies for the user, expand the row of the policy that you want to view.
6. In the policy summary list of services, choose the name of the service that you want to view.

Note

If the policy that you select is an inline policy that is attached directly to the user, then the service summary table appears. If the policy is an inline policy attached from a group, then you are taken to the JSON policy document for that group. If the policy is a managed policy, then you are taken to the service summary for that policy on the **Policies** page.

7. In the service summary list of actions, choose the name of the action that you want to view.

To view the action summary for a policy attached to a role

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, choose **Roles**.
3. In the list of roles, choose the name of the role whose policy you want to view.
4. On the **Summary** page for the role, view the **Permissions** tab to see the list of policies that are attached to the role.
5. In the table of policies for the role, expand the row of the policy that you want to view.

6. In the policy summary list of services, choose the name of the service that you want to view.
7. In the service summary list of actions, choose the name of the action that you want to view.

Understanding the Elements of an Action Summary

The example below is the action summary for the `PutObject` (Write) action from the Amazon S3 service summary (see [Service Summary \(List of Actions\) \(p. 358\)](#)). For this action, the policy defines multiple conditions on a single resource.

The screenshot shows the AWS IAM Action Summary page for the `S3:PutObject` action. At the top, there's a back link (`< Back`), the service and action name (`S3 : PutObject`), and three buttons: `Policy summary`, `{ } JSON`, and `Edit policy`. Below these are a search bar (`Q Filter`) and a filter button (`Showing 1 result`). The main content area has four columns: **Resource** (BucketName = developer_bucket, ObjectPath = All), **Region** (All regions), **Account** (All accounts), and **Request condition** (`s3:x-amz-acl = public-read`). Red numbered circles (1 through 7) point to specific elements: 1 points to the service and action name; 2 points to the `{ } JSON` button; 3 points to the search bar; 4 points to the **Resource** column; 5 points to the **Region** column; 6 points to the **Account** column; and 7 points to the **Request condition** column.

The action summary page includes the following information:

1. Next to the **Back** link appears the name of the service and action in the format `service: action` (in this case **S3: PutObject**). The action summary for this service includes the list of resources that are defined in the policy.
2. Choose `{ } JSON` to see additional details about the policy, such as viewing the multiple conditions that are applied to the actions. (If you are viewing the action summary for an inline policy that is attached directly to a user, the steps differ. To access the JSON policy document in that case, you must close the action summary dialog box and return to the policy summary.)
3. To view the summary for a specific resource, type keywords into the search box to reduce the list of available resources.
4. **Resource** – This column lists the resources that the policy defines for the chosen service. In this example, the **PutObject** action is allowed on all object paths, but on only the `developer_bucket` Amazon S3 bucket resource. Depending on the information that the service provides to IAM, you might see an ARN such as `arn:aws:s3:::developer_bucket/*`, or you might see the defined resource type, such as `BucketName = developer_bucket, ObjectPath = All`.
5. **Region** – This column shows the region in which the resource is defined. Resources can be defined for all regions, or a single region. They cannot exist in more than one specific region.
 - **All regions** – The actions that are associated with the resource apply to all regions. In this example, the action belongs to a global service, Amazon S3. Actions that belong to global services apply to all regions.
 - Region text – The actions associated with the resource apply to one region. For example, a policy can specify the `us-east-2` region for a resource.
6. **Account** – This column indicates whether the services or actions associated with the resource apply to a specific account. Resources can exist in all accounts or a single account. They cannot exist in more than one specific account.
 - **All accounts** – The actions that are associated with the resource apply to all accounts. In this example, the action belongs to a global service, Amazon S3. Actions that belong to global services apply to all accounts.
 - **This account** – The actions that are associated with the resource apply only to the account that you are currently logged in to.
 - Account number – The actions that are associated with the resource apply to one account (one that you are not currently logged in to). For example, if a policy specifies the `123456789012` account for a resource, then the account number appears in the policy summary.

7. **Request condition** – This column shows whether the actions that are associated with the resource are subject to conditions. This example includes the `s3:x-amz-acl = public-read` condition. To learn more about those conditions, choose `{ } JSON` to review the JSON policy document.

Examples of Policy Summaries

The following examples include JSON policies with their associated [policy summaries](#) (p. 348), the [service summaries](#) (p. 358), and the [action summaries](#) (p. 362) to help you understand the permissions given through a policy.

Policy 1: DenyCustomerBucket

This policy demonstrates an allow and a deny for the same service.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccess",
            "Effect": "Allow",
            "Action": ["s3:*"],
            "Resource": ["*"]
        },
        {
            "Sid": "DenyCustomerBucket",
            "Action": ["s3:*"],
            "Effect": "Deny",
            "Resource": ["arn:aws:s3:::customer", "arn:aws:s3:::customer/*"]
        }
    ]
}
```

DenyCustomerBucket Policy Summary:

This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition. For details, choose Show remaining. [Learn more](#)

Policy summary			
{ } JSON		Edit policy	Simulate policy
<input type="text"/> Filter			
Service	Access level	Resource	Request condition
Explicit deny (1 of 103 services)			
S3	Full: Read, Write, Permissions management Limited: List	Multiple	None
Allow (1 of 103 services) Show remaining 102			
S3	Full access	All resources	None

DenyCustomerBucket S3 (Explicit deny) Service Summary:

Action (66 of 69) Hide remaining 3	Resource	Request condition
List (1 of 4 actions)		
HeadBucket (No access)	⚠ This action does not support resource-level permissions. This requires a wildcard (*) for the resource.	None
ListAllMyBuckets(No access)	⚠ This action does not support resource-level permissions. This requires a wildcard (*) for the resource.	None
ListBucket	BucketName = customer	None
ListObjects (No access)	⚠ This action does not support resource-level permissions. This requires a wildcard (*) for the resource.	None
Read (30 of 30 actions)		
GetAccelerateConfiguration	BucketName = customer	None
GetAnalyticsConfiguration	BucketName = customer	None
GetBucketAcl	BucketName = customer	None
GetBucketCORS	BucketName = customer	None
GetBucketLocation	BucketName = customer	None
GetBucketLogging	BucketName = customer	None
GetBucketNotification	BucketName = customer	None
GetBucketPolicy	BucketName = customer	None
GetBucketRequestPayment	BucketName = customer	None
GetBucketTagging	BucketName = customer	None
GetBucketVersioning	BucketName = customer	None
GetBucketWebsite	BucketName = customer	None
GetInventoryConfiguration	BucketName = customer	None
GetIpConfiguration	BucketName = customer	None
GetLifecycleConfiguration	BucketName = customer	None
GetMetricsConfiguration	BucketName = customer	None
GetObject	BucketName = customer, ObjectPath = All	None
GetObjectAcl	BucketName = customer, ObjectPath = All	None
GetObjectTagging	BucketName = customer, ObjectPath = All	None
GetObjectTorrent	BucketName = customer, ObjectPath = All	None
GetObjectVersion	BucketName = customer, ObjectPath = All	None

GetObject (Read) Action Summary:

Resource	Region	Account	Request condition
BucketName = customer, ObjectPath = All	All regions	All accounts	None

Policy 2: DynamoDbRowCognitoID

This policy provides row-level access to Amazon DynamoDB based on the user's Amazon Cognito ID.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb>DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-1:123456789012:table/myDynamoTable"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:LeadingKeys": [
                        "${cognito-identity.amazonaws.com:sub}"
                    ]
                }
            }
        }
    ]
}
```

DynamoDbRowCognitoID Policy Summary:

Service	Access level	Resource	Request condition
Allow (1 of 102 services) Show remaining 101			
DynamoDB	Limited: Read, Write	TableName = myDynamoTable	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}

DynamoDbRowCognitoID DynamoDB (Allow) Service Summary:

Action (4 of 25) Show remaining 21	Resource	Request condition
Read (1 of 14 actions)		
GetItem	TableName = myDynamoTable	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}
Write (3 of 10 actions)		
DeleteItem	TableName = myDynamoTable	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}
PutItem	TableName = myDynamoTable	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}
UpdateItem	TableName = myDynamoTable	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}

GetItem (List) Action Summary:

Resource	Region	Account	Request
TableName = myDynamoTable	us-west-1	123456789012	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}

Policy 3: MultipleResourceCondition

This policy includes multiple resources and conditions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": ["arn:aws:s3:::Apple_bucket/*"],
            "Condition": {"StringEquals": {"s3:x-amz-acl": ["public-read"]}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": ["arn:aws:s3:::Orange_bucket/*"],
            "Condition": {"StringEquals": {
                "s3:x-amz-acl": ["custom"],
                "s3:x-amz-grant-full-control": ["1234"]
            }}
        }
    ]
}
```

MultipleResourceCondition Policy Summary:

Service	Access level	Resource	Request condition
Allow (1 of 100 services) Show remaining 99			
S3	Limited: Write, Permissions management	Multiple	Multiple

MultipleResourceCondition S3 (Allow) Service Summary:

Action (2 of 52 actions)	Show remaining 50	Resource	Request condition
Write (1 of 21 actions)			
PutObject		Multiple	Multiple
Permissions management (1 of 5 actions)			
PutObjectAcl		Multiple	Multiple

PutObject (Write) Action Summary:

Resource	Region	Account	Request condition
BucketName = Orange_bucket, ObjectPath = All	All regions	All accounts	Multiple
BucketName = Apple_bucket, ObjectPath = All	All regions	All accounts	s3:x-amz-acl = public-read

Policy 4: EC2_Troubleshoot

The following policy allows users to get a screenshot of a running Amazon EC2 instance, which can help with EC2 troubleshooting. This policy also permits viewing information about the items in the Amazon S3 developer bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:GetConsoleScreenshot"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3>ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::developer"
            ]
        }
    ]
}
```

EC2_Troubleshoot Policy Summary:

Service	Access level	Resource	Request condition
Allow (2 of 102 services) Show remaining 100			
EC2	Limited: Read	All resources	None
S3	Limited: List	BucketName = developer	None

EC2_Troubleshoot S3 (Allow) Service Summary:

Action (1 of 52) Show remaining 51	Resource	Request condition
List (1 of 4 actions)		
ListBucket	BucketName = developer	None

ListBucket (List) Action Summary:

<input type="text"/> Filter			
Resource	Region	Account	Request
BucketName = developer	All regions	All accounts	None

Policy 5: Unrecognized_Service_Action

The following policy was intended to provide full access to DynamoDB, but that access fails because dynamodb is misspelled as dynamobd. This policy was intended to allow access to some Amazon EC2 actions in the us-east-2 region, but deny that access to the ap-northeast-2 region. However, access

to reboot instances in the `ap-northeast-2` region is not explicitly denied because of the unrecognized `o` in the middle of the `RebootInstances` action. This example shows how you can use policy summaries to locate errors in your policies. To learn how to edit policies based on information in a policy summary, see [Editing Policies to Fix Warnings \(p. 350\)](#).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:*"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Action": [
                "ec2:RunInstances",
                "ec2:StartInstances",
                "ec2:StopInstances",
                "ec2:RebootInstances"
            ],
            "Resource": "*",
            "Effect": "Deny",
            "Condition": {
                "StringEquals": {
                    "ec2:Region": "ap-northeast-2"
                }
            }
        },
        {
            "Action": [
                "ec2:RunInstances",
                "ec2:StartInstances",
                "ec2:StopInstances",
                "ec2:RebootInstances"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Condition": {
                "StringEquals": {
                    "ec2:Region": "us-east-2"
                }
            }
        }
    ]
}
```

Unrecognized_Service_Action Policy Summary:

Service	Access level	Resource	Request condition
<u>Unrecognized services</u>			
dynamodb	⚠		
<u>Explicit deny (1 of 103 services)</u>			
EC2	⚠ Limited: Write	All resources	ec2:Region = ap-northeast-2
<u>Allow (1 of 103 services)</u> Show remaining 102			
EC2	⚠ Limited: Write	All resources	ec2:Region = us-east-2

Unrecognized_Service_Action EC2 (Explicit deny) Service Summary:

Action (3 of 229) Show remaining 226	Resource	Request condition
Unrecognized actions		
RebootInstances ⚠		
Write (3 of 157 actions)		
RunInstances	All resources	ec2:Region = ap-northeast-2
StartInstances	All resources	ec2:Region = ap-northeast-2
StopInstances	All resources	ec2:Region = ap-northeast-2

Unrecognized_Service_Action StartInstances (Write) Action Summary:

Resource	Region	Account	Request condition
All resources	All regions	All accounts	ec2:Region = ap-northeast-2

Policy 6: CodeBuild_CodeCommit_CodeDeploy

This policy provides access to specific CodeBuild, CodeCommit, and CodeDeploy resources. Because these resources are specific to each service, they appear only with the matching service. If you include a resource that does not match any services in the `Action` element, then the resource appears in all action summaries.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1487980617000",
            "Effect": "Allow",
            "Action": [
                "codebuild:*",
                "codecommit:*",
                "codedeploy:*"
            ],
            "Resource": [
                "arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project",
                "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo",
                "arn:aws:codedeploy:us-east-2:123456789012:application:WordPress_App",
                "arn:aws:codedeploy:us-east-2:123456789012:instance/AssetTag*"
            ]
        }
    ]
}
```

CodeBuild_CodeCommit_CodeDeploy Policy Summary:

Service	Access level	Resource	Request condition
Allow (3 of 103 services) Show remaining 100			
CodeBuild	Limited: List, Read, Write	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
CodeCommit	Full: Read, Write Limited: List	arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo	None
CodeDeploy	Limited: List, Read, Write	Multiple	None

CodeBuild_CodeCommit_CodeDeploy CodeBuild (Allow) Service Summary:

Action (9 of 15) Show remaining 6	Resource	Request condition
List (1 of 3 actions)		
ListBuildsForProject	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
Read (2 of 5 actions)		
BatchGetBuilds	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
BatchGetProjects	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
Write (6 of 7 actions)		
BatchDeleteBuilds	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
CreateProject	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
DeleteProject	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
StartBuild	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
StopBuild	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
UpdateProject	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None

CodeBuild_CodeCommit_CodeDeploy StartBuild (Write) Action Summary:

Resource	Region	Account	Request condition
arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	us-east-2	123456789012	None

Permissions Required to Access IAM Resources

Resources are objects within a service. IAM resources include groups, users, roles, and policies. If you are signed in with AWS account root user credentials, you have no restrictions on administering IAM.

credentials or IAM resources. However, IAM users must explicitly be given permissions to administer credentials or IAM resources. You can do this by attaching an identity-based policy to the user.

Note

Throughout the AWS documentation, when we refer to an IAM policy without mentioning any of the specific categories, we mean an identity-based, customer managed policy. For details about policy categories, see [the section called "Policies" \(p. 275\)](#).

Permissions for Administering IAM Identities

The permissions that are required to administer IAM groups, users, roles, and credentials usually correspond to the API actions for the task. For example, in order to create IAM users, you must have the `iam:CreateUser` permission that has the corresponding API command: `CreateUser`. To allow an IAM user to create other IAM users, you could attach an IAM policy like the following one to that user:

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Action": "iam:CreateUser",  
        "Resource": "*"  
    }  
}
```

In a policy, the value of the `Resource` element depends on the action and what resources the action can affect. In the preceding example, the policy allows a user to create any user (* is a wildcard that matches all strings). In contrast, a policy that allows users to change only their own access keys (API actions `CreateAccessKey` and `UpdateAccessKey`) typically has a `Resource` element. In that case the ARN includes a variable that resolves to the current user's name, as in the following example (replace ACCOUNT-ID-WITHOUT-HYPHENS with your AWS account ID):

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Action": [  
            "iam:CreateAccessKey",  
            "iam:UpdateAccessKey"  
        ],  
        "Resource": "arn:aws:iam::accountid:user/${aws:username}"  
    }  
}
```

In the previous example, `${aws:username}` is a variable that resolves to the user name of the current user. For more information about policy variables, see [IAM Policy Elements: Variables \(p. 451\)](#).

Using a wildcard character (*) in the action name often makes it easier to grant permissions for all the actions related to a specific task. For example, to allow users to perform any IAM action, you can use `iam:*` for the action. To allow users to perform any action related just to access keys, you can use `iam:*AccessKey*` in the `Action` element of a policy statement. This gives the user permission to perform the `CreateAccessKey`, `DeleteAccessKey`, `GetAccessKeyLastUsed`, `ListAccessKeys`, and `UpdateAccessKey` actions. (If an action is added to IAM in the future that has "AccessKey" in the name, using `iam:*AccessKey*` for the `Action` element will also give the user permission to that new action.) The following example shows a policy that allows users to perform all actions pertaining to their own access keys (replace ACCOUNT-ID-WITHOUT-HYPHENS with your AWS account ID):

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

```
        "Effect": "Allow",
        "Action": "iam:*AccessKey*",
        "Resource": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:user/${aws:username}"
    }
}
```

Some tasks, such as deleting a group, involve multiple actions: You must first remove users from the group, then detach or delete the group's policies, and then actually delete the group. If you want a user to be able to delete a group, you must be sure to give the user permissions to perform all of the related actions.

Permissions for Working in the AWS Management Console

The preceding examples show policies that allow a user to perform the actions with the [AWS CLI](#) or the [AWS SDKs](#).

As users work with the console, the console issues requests to IAM to list groups, users, roles, and policies, and to get the policies associated with a group, user, or role. The console also issues requests to get AWS account information and information about the principal. The principal is the user making requests in the console.

In general, to perform an action, you must have only the matching action included in a policy. To create a user, you need permission to call the [CreateUser](#) action. Often, when you use the console to perform an action, you must have permissions to display, list, get, or otherwise view resources in the console. This is necessary so that you can navigate through the console to make the specified action. For example, if user Jorge wants to use the console to change his own access keys, he goes to the IAM console and chooses [Users](#). This action causes the console to make a [ListUsers](#) request. If Jorge doesn't have permission for the [iam>ListUsers](#) action, the console is denied access when it tries to list users. As a result, Jorge can't get to his own name and to his own access keys, even if he has permissions for the [CreateAccessKey](#) and [UpdateAccessKey](#) actions.

For example, if user Bob wants to use the console to change his own access keys, he goes to the IAM console and chooses [Users](#). This action causes the console to make a [ListUsers](#) request. If Bob doesn't have permission for the [iam>ListUsers](#) action, the console is denied access when it tries to list users. As a result, Bob can't get to his own name and to his own access keys, even if he has permissions for the [CreateAccessKey](#) and [UpdateAccessKey](#) actions.

If you want to give users permissions to administer groups, users, roles, policies, and credentials with the AWS Management Console, you need to include permissions for the actions that the console performs. For some examples of policies that you can use to grant a user for these permissions, see [Example Policies for Administering IAM Resources \(p. 376\)](#).

Granting Permissions Across AWS Accounts

You can directly grant IAM users in your own account access to your resources. If users from another account need access to your resources, you can create an IAM role, which is an entity that includes permissions but that isn't associated with a specific user. Users from other accounts can then use the role and access resources according to the permissions you've assigned to the role. For more information, see [Providing Access to an IAM User in Another AWS Account That You Own \(p. 138\)](#).

Note

For services that support resource-based policies as described in [Identity-Based Policies and Resource-Based Policies \(p. 285\)](#) (such as Amazon S3, Amazon SNS, and Amazon SQS), an alternative to using roles is to attach a policy to the resource (bucket, topic, or queue) that you want to share. The resource-based policy can specify the AWS account that has permissions to access the resource.

Permissions for One Service to Access Another

Many AWS services access other AWS services. For example, several AWS services—including Amazon EMR, Elastic Load Balancing, and Amazon EC2 Auto Scaling—manage Amazon EC2 instances. Other AWS services make use of Amazon S3 buckets, Amazon SNS topics, Amazon SQS queues, and so on.

The scenario for managing permissions in these cases varies by service. Here are some examples of how permissions are handled for different services:

- In Amazon EC2 Auto Scaling, users must have permission to use Auto Scaling, but don't need to be explicitly granted permission to manage Amazon EC2 instances.
- In AWS Data Pipeline, an IAM role determines what a pipeline can do; users need permission to assume the role. (For details, see [Granting Permissions to Pipelines with IAM](#) in the *AWS Data Pipeline Developer Guide*.)

For details about how to configure permissions properly so that an AWS service is able to accomplish the tasks you intend, refer to the documentation for the service you are calling. To learn how to create a role for a service, see [Creating a Role to Delegate Permissions to an AWS Service \(p. 191\)](#).

Configuring a service with an IAM role to work on your behalf

When you want to configure an AWS service to work on your behalf, you typically provide the ARN for an IAM role that defines what the service is allowed to do. AWS checks to ensure that you have permissions to pass a role to a service. For more information, see [Granting a User Permissions to Pass a Role to an AWS Service \(p. 207\)](#).

Required Actions

Actions are the things that you can do to a resource, such as viewing, creating, editing, and deleting that resource. Actions are defined by each AWS service.

To allow someone to perform an action, you must include the necessary actions in a policy that applies to the calling identity or the affected resource. In general, to provide the permission required to perform an action, you must include that action in your policy. For example, to create a user, you need add the `CreateUser` action to your policy.

In some cases, an action might require that you include additional related actions in your policy. For example, to provide permission for someone to create a directory in AWS Directory Service using the `ds:CreateDirectory` operation, you must include the following actions in their policy:

- `ds:CreateDirectory`
- `ec2:DescribeSubnets`
- `ec2:DescribeVpcs`
- `ec2:CreateSecurityGroup`
- `ec2:CreateNetworkInterface`
- `ec2:DescribeNetworkInterfaces`
- `ec2:AuthorizeSecurityGroupIngress`
- `ec2:AuthorizeSecurityGroupEgress`

When you create or edit a policy using the visual editor, you receive warnings and prompts to help you choose all of the required actions for your policy.

For more information about the permissions required to create a directory in AWS Directory Service, see [Example 2: Allow a User to Create a Directory](#).

Example Policies for Administering IAM Resources

Following are examples of IAM policies that allow users to perform tasks associated with managing IAM users, groups, and credentials. This includes policies that permit users manage their own passwords, access keys, and multi-factor authentication (MFA) devices.

For examples of policies that let users perform tasks with other AWS services, like Amazon S3, Amazon EC2, and DynamoDB, see [Example Policies \(p. 295\)](#).

Topics

- [Allow Users to Manage Their Own Passwords \(from the My Password Page\) \(p. 376\)](#)
- [Allow Users to Manage Their Own Passwords, Access Keys, and SSH Keys \(p. 377\)](#)
- [Allow a User to List the Account's Groups, Users, Policies, and More for Reporting Purposes \(p. 378\)](#)
- [Allow a User to Manage a Group's Membership \(p. 378\)](#)
- [Allow a User to Manage IAM Users \(p. 378\)](#)
- [Allow Users to Set Account Password Policy \(p. 379\)](#)
- [Allow Users to Generate and Retrieve IAM Credential Reports \(p. 380\)](#)
- [Allow Users to Manage Only Their Own Virtual MFA Devices \(p. 380\)](#)
- [Allow All IAM Actions \(Admin Access\) \(p. 381\)](#)

Allow Users to Manage Their Own Passwords (from the My Password Page)

If the account's [password policy \(p. 75\)](#) is set to allow all users to change their own passwords, you don't need to attach any permissions to individual users or groups. All users are able to go to the [My Password page \(p. 83\)](#) in the AWS Management Console that lets them change their own password.

If the account's password policy is *not* set to allow all users to change their own passwords, you can attach the following policy to selected users or groups to allow those users to change only their own passwords. This policy only allows users to use the special [My Password page](#) in the console; it does not give users permissions to work through the dashboard in the IAM console.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:GetAccountPasswordPolicy",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:ChangePassword",
      "Resource": "arn:aws:iam::account-id-without-hyphens:user/${aws:username}"
    }
  ]
}
```

If users do not use the console to change their own password, they do not need the `iam:GetAccountPasswordPolicy` permission. They can instead run the `aws iam change-password` command from the AWS CLI, or make a request with the `ChangePassword` action.

For information about letting selected users manage passwords using the **Users** section of the IAM console, see the next section.

Allow Users to Manage Their Own Passwords, Access Keys, and SSH Keys

The following policy allows users to perform these actions in the AWS Management Console:

- Create, change, or remove their own password. This includes the [CreateLoginProfile](#), [DeleteLoginProfile](#), [GetLoginProfile](#), and [UpdateLoginProfile](#) actions.
- Create or delete their own access key (access key ID and secret access key). This includes the [CreateAccessKey](#), [DeleteAccessKey](#), [GetAccessKeyLastUsed](#), [ListAccessKeys](#), and [UpdateAccessKey](#) actions.
- Create or delete their own SSH keys. This includes the [UploadSSHPublicKey](#), [DeleteSSHPublicKey](#), [GetSSHPublicKey](#), [ListSSHPublicKeys](#), and [UpdateSSHPublicKey](#) actions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:*LoginProfile",
                "iam:*AccessKey*",
                "iam:*SSHPublicKey"
            ],
            "Resource": "arn:aws:iam::account-id-without-hyphens:user/${aws:username}"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam>ListAccount*",
                "iam:GetAccountSummary",
                "iam:GetAccountPasswordPolicy",
                "iam>ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

The actions in the preceding policy include wildcards (for example, `iam:*LoginProfile`, `iam:*AccessKey*`, and `iam:*SSHPublicKey*`). This is a convenient way to include a set of related actions. If you want to remove permissions for any one of the related actions, you must instead list each of the individual actions. For example, if you don't want users to be able to delete a password, you must individually list `iam>CreateLoginProfile`, `iam:DeleteLoginProfile`, and `iam:UpdateLoginProfile`, and omit `iam:DeleteLoginProfile`.

The second element in the Statement array, including `iam:GetAccountSummary`, `iam:GetAccountPasswordPolicy`, `iam>ListAccount*`, and `iam>ListUsers` permissions, allows the user to see certain information on the IAM console dashboard, such as whether a password policy is enabled, how many groups the account has, what the account URL and alias are, etc. For example, the [GetAccountSummary](#) action returns an object that contains a collection of information about the account that is then displayed on the IAM console dashboard.

The following policy is like the previous one but excludes the permissions that are needed only for console access. This policy lets users manage their credentials with the [AWS CLI](#), Tools for Windows PowerShell, the [AWS SDKs](#), or the IAM HTTP query API.

```
{
```

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Action": [  
            "iam:*LoginProfile",  
            "iam:*AccessKey*",  
            "iam:*SSHPublicKey*"  
        ],  
        "Resource": "arn:aws:iam::account-id-without-hyphens:user/${aws:username}"  
    }  
}
```

Allow a User to List the Account's Groups, Users, Policies, and More for Reporting Purposes

The following policy allows the user to call any IAM action that starts with the string `Get` or `List`.

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

The benefit of using `Get*` and `List*` actions is that if new types of entities are added to IAM in the future, the access granted in the policy to `Get*` and `List*` all actions would automatically allow the user to list those new entities.

Allow a User to Manage a Group's Membership

The following policy allows the user to update the membership of the group called `MarketingGroup`. To use the following policy, replace `ACCOUNT-ID-WITHOUT-HYPHENS` with your AWS account ID.

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Action": [  
            "iam:AddUserToGroup",  
            "iam:RemoveUserFromGroup",  
            "iam:GetGroup"  
        ],  
        "Resource": "arn:aws:iam::account-id-without-hyphens:group/MarketingGroup"  
    }  
}
```

Allow a User to Manage IAM Users

The following policy allows a user to perform all the tasks associated with managing IAM users but not to perform actions on other entities, such as creating groups or policies. Allowed actions include these:

- Creating the user (the `CreateUser` action).
- Deleting the user. This task requires permissions to perform all of the following actions: `DeleteSigningCertificate`, `DeleteLoginProfile`, `RemoveUserFromGroup`, and `DeleteUser`.

- Listing users in the account and in groups (the [GetUser](#), [ListUsers](#) and [ListGroupsForUser](#) actions).
- Listing and removing policies for the user (the [ListUserPolicies](#), [ListAttachedUserPolicies](#), [DetachUserPolicy](#), [DeleteUserPolicy](#) actions)
- Renaming or changing the path for the user (the [UpdateUser](#) action). The `Resource` element must include an ARN that covers both the source path and the target path. For more information on paths, see [Friendly Names and Paths \(p. 410\)](#).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUsersToPerformUserActions",
            "Effect": "Allow",
            "Action": [
                "iam:CreateUser",
                "iam>ListUsers",
                "iam:GetUser",
                "iam:UpdateUser",
                "iam>DeleteUser",
                "iam>ListGroupsForUser",
                "iam>ListUserPolicies",
                "iam>ListAttachedUserPolicies",
                "iam>DeleteSigningCertificate",
                "iam>DeleteLoginProfile",
                "iam:RemoveUserFromGroup",
                "iam:DetachUserPolicy",
                "iam>DeleteUserPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUsersToSeeStatsOnIAMConsoleDashboard",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccount*",
                "iam>ListAccount*"
            ],
            "Resource": "*"
        }
    ]
}
```

A number of the permissions included in the preceding policy allow the user to perform tasks in the AWS Management Console. Users who perform user-related tasks from the [AWS CLI](#), the [AWS SDKs](#), or the IAM HTTP query API only might not need certain permissions. For example, if users already know the ARN of policies to detach from a user, they do not need the `iam>ListAttachedUserPolicies` permission. The exact list of permissions that a user requires depends on the tasks that the user must perform while managing other users.

The following permissions in the policy allow access to user tasks via the AWS Management Console:

- `iam:GetAccount*`
- `iam>ListAccount*`

Allow Users to Set Account Password Policy

You might give some users permissions to get and update your AWS account's [password policy \(p. 75\)](#). The following example policy grants these permissions.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        "Effect": "Allow",  
        "Action": [  
            "iam:GetAccountPasswordPolicy",  
            "iam:UpdateAccountPasswordPolicy"  
        ],  
        "Resource": "*"  
    ]  
}
```

Allow Users to Generate and Retrieve IAM Credential Reports

You can give users permission to generate and download a report that lists all users in your AWS account and the status of their various credentials, including passwords, access keys, MFA devices, and signing certificates. The following example policy grants these permissions.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        "Effect": "Allow",  
        "Action": [  
            "iam:GenerateCredentialReport",  
            "iam:GetCredentialReport"  
        ],  
        "Resource": "*"  
    ]  
}
```

For more information about credential reports, see [Getting Credential Reports for Your AWS Account \(p. 120\)](#).

Allow Users to Manage Only Their Own Virtual MFA Devices

A [virtual MFA device \(p. 93\)](#) is a software implementation of a device that provides one-time passwords. Virtual MFA devices are hosted on a physical hardware device (typically a smartphone). In order to configure a virtual MFA device, you must have access to the physical device where the virtual MFA device is hosted. If your users create virtual MFA devices inside a smartphone app on their own smartphone, you might want to let them configure the devices themselves. For more about using virtual MFA devices with IAM, see [Enabling a Virtual Multi-factor Authentication \(MFA\) Device \(p. 93\)](#).

The following policy allows a user to configure his or her own virtual MFA device from the AWS Management Console or using any of the command-line tools. The policy allows only MFA-authenticated users to deactivate and delete their own virtual MFA devices.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowUsersToCreateEnableResyncDeleteTheirOwnVirtualMFADevice",  
            "Effect": "Allow",  
            "Action": [  
                "iam>CreateVirtualMFADevice",  
                "iam:EnableMFADevice",  
                "iam:ResyncMFADevice",  
                "iam>DeleteVirtualMFADevice"  
            ],  
            "Resource": [  
                "arn:aws:iam::  
                    AWS account number:virtualMFADevice/*  
            ]  
        }  
    ]  
}
```

```

        "arn:aws:iam::account-id-without-hyphens:mfa/${aws:username}",
        "arn:aws:iam::account-id-without-hyphens:user/${aws:username}"
    ],
},
{
    "Sid": "AllowUsersToDeactivateTheirOwnVirtualMFADevice",
    "Effect": "Allow",
    "Action": [
        "iam:DeactivateMFADevice"
    ],
    "Resource": [
        "arn:aws:iam::account-id-without-hyphens:mfa/${aws:username}",
        "arn:aws:iam::account-id-without-hyphens:user/${aws:username}"
    ],
    "Condition": {
        "Bool": {
            "aws:MultiFactorAuthPresent": true
        }
    },
    {
        "Sid": "AllowUsersToListMFADevicesandUsersForConsole",
        "Effect": "Allow",
        "Action": [
            "iam>ListMFADevices",
            "iam>ListVirtualMFADevices",
            "iam>ListUsers"
        ],
        "Resource": "*"
    }
]
}

```

Note

The action `iam>DeleteVirtualMFADevice` is *not* subject to the MFA condition check because it is included in the first statement instead of the second. This isn't a security concern because you can only delete an MFA device after you deactivate it, which the user can do only if they are MFA authenticated. This prevents a situation that can occur if you cancel the **Create MFA Device** wizard after it creates the device but before it validates the two codes and associates it with the user. Because the user is not yet MFA authenticated at this point, the wizard (which operates with the user's permissions) fails to clean up the device if the policy requires MFA authentication to delete the device.

Allow All IAM Actions (Admin Access)

You might give some users administrative permissions to perform all actions in IAM, including managing passwords, access keys, MFA devices, and user certificates. The following example policy grants these permissions.

Warning

When you give a user full access to IAM, there is no limit to the permissions that user can grant to him/herself or others. The user can create new IAM entities (users or roles) and grant those entities full access to all resources in your AWS account. When you give a user full access to IAM, you are effectively giving them full access to all resources in your AWS account. This includes access to delete all resources. You should grant these permissions to only trusted administrators, and you should enforce multi-factor authentication (MFA) for these administrators.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "iam:*",
    }
}
```

```
        "Resource": "*"  
    }  
}
```

Troubleshooting IAM

If you encounter access-denied issues or similar difficulties when working with AWS Identity and Access Management (IAM), consult the topics in this section.

Topics

- [Troubleshooting General Issues \(p. 383\)](#)
- [Troubleshoot IAM Policies \(p. 386\)](#)
- [Troubleshooting IAM Roles \(p. 399\)](#)
- [Troubleshooting Amazon EC2 and IAM \(p. 401\)](#)
- [Troubleshooting Amazon S3 and IAM \(p. 404\)](#)
- [Troubleshooting SAML 2.0 Federation with AWS \(p. 405\)](#)

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 Identity and Access Management (IAM).

Topics

- [I lost my access keys \(p. 383\)](#)
- [I need to access an old account \(p. 383\)](#)
- [I get "access denied" when I make a request to an AWS service \(p. 384\)](#)
- [I get "access denied" when I make a request with temporary security credentials \(p. 384\)](#)
- [Policy variables aren't working \(p. 385\)](#)
- [Changes that I make are not always immediately visible \(p. 385\)](#)

I lost my access keys

Access keys consist of two parts:

- **The access key identifier.** This is not a secret, and can be seen in the IAM console wherever access keys are listed, such as on the user summary page.
- **The secret access key.** This is provided when you initially create the access key pair. Just like a password, it ***cannot be retrieved later***. If you lost your secret access key, then you must create a new access key pair. If you already have the [maximum number of access keys \(p. 415\)](#), you must delete an existing pair before you can create another.

For more information, see [Resetting Your Lost or Forgotten Passwords or Access Keys \(p. 90\)](#).

I need to access an old account

When you first created your AWS account, you provided an email address and password. These are your AWS account root user credentials. If you have an old AWS account that you can no longer access because you have lost or forgotten the password, you can recover the password. For more information, see [Resetting Your Lost or Forgotten Passwords or Access Keys \(p. 90\)](#).

If you no longer have access to the email, you should first try to recover access to the email. If that doesn't work, you can contact AWS Customer Service.

You can try to recover access to your email using one of the following options:

- If you own the domain where the email address is hosted, you can add the email address back to your email server. Alternatively, you can set up a catch-all for your email account. The catch-all setting on your domain "catches all" messages that are sent to email addresses that do not exist in the mail server. It redirects those messages to a specific email address. For example, if your AWS account root user email address is paulo@sample-domain.com, but you changed your only domain email address to paulo.santos@sample-domain.com, then you could set your new email as the catch-all. That way, when someone like AWS sends a message to paulo@sample-domain.com or any other text@sample-domain.com, you receive that message at paulo.santos@sample-domain.com.
- If the email address on the account is part of your corporate email system, we recommend that you contact your IT system administrators. They might be able to help you regain access to the email address.

If you're still not able to access your AWS account, you can find alternate support options at [Contact us](#) by expanding the **I'm an AWS customer and I'm looking for billing or account support** menu. When you contact Customer Service, you must provide the following information:

- All the details that are listed on the account, including your full name, phone number, address, email address, and last four digits of the credit card. You might need to create a new AWS account for the purpose of contacting Customer Service, but this is necessary to assist in the investigation of your request.
- The reason that you're unable to access the email account to receive password reset instructions
- Also request that the support team delete any accounts that you are not using. It's a good idea not to have open accounts in your name that could result in charges to you.

I get "access denied" when I make a request to an AWS service

Verify that you have permission to call the action and resource that you have requested. If any conditions are set, you must also meet those conditions when you send the request. For information about viewing or modifying policies for an IAM user, group, or role, see [Managing IAM Policies \(p. 316\)](#).

If you're trying to access a service that has [resource-based policies \(p. 285\)](#), such as Amazon S3, Amazon SNS, or Amazon SQS, verify that the policy specifies you as a principal and grants you access. To view the services that support resource-based policies, see [AWS Services That Work with IAM \(p. 417\)](#).

If you are signing requests manually (without using the [AWS SDKs](#)), verify that you have correctly [signed the request](#).

I get "access denied" when I make a request with temporary security credentials

- Verify that the service accepts temporary security credentials, see [AWS Services That Work with IAM \(p. 417\)](#).
- Verify that your requests are being signed correctly and that the request is well-formed. For details, see your [toolkit](#) documentation or [Using Temporary Security Credentials to Request Access to AWS Resources \(p. 242\)](#).
- Verify that your temporary security credentials haven't expired. For more information, see [Temporary Security Credentials \(p. 231\)](#).

- Verify that the IAM user or role has the correct permissions. Permissions for temporary security credentials are derived from an IAM user or role, so the permissions are limited to those granted to the IAM user or role. For more information about how permissions for temporary security credentials are determined, see [Controlling Permissions for Temporary Security Credentials \(p. 246\)](#).
- If you are accessing a resource that has a resource-based policy by using a role, verify that the policy grants permissions to the role. For example, the following policy allows MyRole from account 111122223333 to access MyBucket.

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Sid": "S3BucketPolicy",  
        "Effect": "Allow",  
        "Principal": {"AWS": ["arn:aws:iam::111122223333:role/MyRole"]},  
        "Action": ["s3:PutObject"],  
        "Resource": ["arn:aws:s3:::MyBucket/*"]  
    }]  
}
```

Policy variables aren't working

- Verify that all policies that include variables include the following version number in the policy: "Version": "2012-10-17". Without the correct version number, the variables are not replaced during evaluation. Instead, the variables are evaluated literally. Any policies that don't include variables will still work if you include the latest version number.

A **Version** policy element is different from a policy version. The **Version** policy element is used within a policy and defines the version of the policy language. A policy version, on the other hand, is created when you make changes to a customer managed policy in IAM. The changed policy doesn't overwrite the existing policy. Instead, IAM creates a new version of the managed policy. To learn more about the **Version** policy element see [IAM JSON Policy Elements: Version \(p. 426\)](#). To learn more about policy versions, see [the section called "Versioning IAM Policies" \(p. 335\)](#).

- Verify that your policy variables are in the right case. For details, see [IAM Policy Elements: Variables \(p. 451\)](#).

Changes that I make are not always immediately visible

As a service that is accessed through computers in data centers around the world, IAM uses a distributed computing model called [eventual consistency](#). Any change that you make in IAM (or other AWS services) 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, from replication zone to replication zone, and from region to region around the world. IAM 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.

You must 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 is not instantly visible at another. Such changes include creating or updating users, groups, roles, or policies. We recommend that you do not include such IAM changes in the critical, high-availability code paths of your application. Instead, make IAM changes in a separate initialization or setup routine that you run less frequently. Also, be sure to verify that the changes have been propagated before production workflows depend on them.

For more information about how some other AWS services are affected by this, consult the following resources:

- **Amazon DynamoDB:** [What is the consistency model of Amazon DynamoDB?](#) in the *DynamoDB FAQ*, and [Read Consistency](#) in the Amazon DynamoDB Developer Guide.
- **Amazon EC2:** [EC2 Eventual Consistency](#) in the *Amazon EC2 API Reference*.
- **Amazon EMR:** [Ensuring Consistency When Using Amazon S3 and Amazon Elastic MapReduce for ETL Workflows](#) in the AWS Big Data Blog
- **Amazon Redshift:** [Managing Data Consistency](#) in the *Amazon Redshift Database Developer Guide*
- **Amazon S3:** [Amazon S3 Data Consistency Model](#) in the *Amazon Simple Storage Service Developer Guide*

Troubleshoot IAM Policies

A [policy \(p. 275\)](#) is an entity in AWS that, when attached to an identity or resource, defines their permissions. AWS evaluates these policies when a principal, such as a user, makes a request. Permissions in the policies determine whether the request is allowed or denied. Policies are stored in AWS as JSON documents that are attached to principals as *identity-based policies* or to resources as *resource-based policies*. You can attach an identity-based policy to a principal (or identity), such as an IAM group, user, or role. Identity-based policies include AWS managed policies, customer managed policies, and inline policies. You can create and edit customer managed policies in the AWS Management Console using the **Visual editor** tab or the **JSON** tab. When you view a policy in the AWS Management Console, you can see a summary of the permissions that are granted by that policy. You can use the visual editor and policy summaries to help you diagnose and fix common errors encountered while managing IAM policies.

Keep in mind that all IAM policies are stored using syntax that begins with the rules of [JavaScript Object Notation \(JSON\)](#). You do not have to understand this syntax to create or manage your policies. You can create and edit a policy using the visual editor in the AWS Management Console. To learn more about JSON syntax in IAM policies, see [Grammar of the IAM JSON Policy Language \(p. 463\)](#).

Troubleshooting IAM Policy Topics

- [Troubleshoot Using the Visual Editor \(p. 387\)](#)
 - [Policy Restructuring \(p. 387\)](#)
 - [Choosing a Resource ARN in the Visual Editor \(p. 387\)](#)
 - [Denying Permissions in the Visual Editor \(p. 388\)](#)
 - [Specifying Multiple Services in the Visual Editor \(p. 388\)](#)
 - [Reducing the Size of Your Policy in the Visual Editor \(p. 388\)](#)
 - [Fixing Unrecognized Services, Actions, or Resource Types in the Visual Editor \(p. 389\)](#)
- [Troubleshoot Using Policy Summaries \(p. 390\)](#)
 - [Missing Policy Summary \(p. 390\)](#)
 - [Policy Summary Includes Unrecognized Services, Actions, or Resource Types \(p. 390\)](#)
 - [Service Does Not Support IAM Policy Summaries \(p. 391\)](#)
 - [My Policy Does Not Grant the Expected Permissions \(p. 392\)](#)
- [Troubleshoot Policy Management \(p. 396\)](#)
 - [Attaching or Detaching a Policy in an IAM Account \(p. 396\)](#)
- [Troubleshoot JSON Policy Documents \(p. 396\)](#)
 - [More Than One JSON Policy Object \(p. 396\)](#)
 - [More Than One JSON Statement Element \(p. 397\)](#)
 - [More Than One Effect, Action, or Resource Element in a JSON Statement Element \(p. 397\)](#)
 - [Missing JSON Version Element \(p. 399\)](#)

Troubleshoot Using the Visual Editor

When you create or edit a customer managed policy, you can use information in the **Visual editor** tab to help you troubleshoot errors in your policy. To view an example of using the visual editor to create a policy, see the section called “[Controlling Access to Identities](#)” (p. 288).

Policy Restructuring

When you create a policy, AWS validates, processes, and transforms the policy before storing it. When AWS returns the policy in response to a user query or displays it in the console, AWS transforms the policy back into a human-readable format without changing the permissions granted by the policy. This can result in differences in what you see in the policy visual editor or **JSON** tab: Visual editor permission blocks can be added, removed, or reordered, and content within a block can be optimized. In the **JSON** tab, insignificant white space can be removed, and elements within JSON maps can be reordered. In addition, AWS account IDs within the principal elements can be replaced by the ARN of the AWS account root user. Because of these possible changes, you should not compare JSON policy documents as strings.

When you create a customer managed policy in the AWS Management Console, you can choose to work entirely in the **JSON** tab. If you never make any changes in the **Visual editor** tab and choose **Review policy** from the **JSON** tab, the policy is less likely to be restructured. However, if you create a policy and use the **Visual editor** tab to make any modifications, or if you choose **Review policy** from the **Visual editor** tab, then IAM might restructure the policy to optimize its appearance in the visual editor.

This restructuring exists only in your editing session and is not saved automatically.

If your policy is restructured in your editing session, IAM determines whether to save the restructuring based on the following situations:

On this tab	If you edit your policy	And then choose <i>Review policy</i> from this tab	When you choose <i>Save changes</i>
Visual editor	Edited	Visual editor	The policy is restructured
Visual editor	Edited	JSON	The policy is restructured
Visual editor	Not Edited	Visual editor	The policy is restructured
JSON	Edited	Visual editor	The policy is restructured
JSON	Edited	JSON	The policy structure is not changed
JSON	Not Edited	JSON	The policy structure is not changed

IAM might restructure complex policies or policies that have permission blocks or statements that allow multiple services, resource types, or condition keys.

Choosing a Resource ARN in the Visual Editor

When you create or edit a policy using the visual editor, you must first choose a service, and then choose actions from that service. If the service and actions that you selected support choosing [specific](#)

resources (p. 294), then the visual editor lists the supported resource types. You can then choose **Add ARN** to provide the details about your resource. You can choose from the following options for adding an ARN for a resource type.

- **Use the ARN builder** – Based on the resource type, you might see different fields to build your ARN. You can also choose **Any** to provide permissions for any value for the specified setting. For example, if you selected the Amazon EC2 **Read** access level group, then the actions in your policy support the **instance** resource type. You must provide the **Region**, **Account**, and **InstanceId** values for your resource. If you provide your account ID but choose **Any** for the region and instance ID, then the policy grants permissions to any instance in your account.
- **Type or paste the ARN** – You can specify resources by their [Amazon Resource Name \(ARN\)](#). You can include a wildcard character (*) in any field of the ARN (between each pair of colons). For more information, see [IAM JSON Policy Elements: Resource](#) (p. 436).

Denying Permissions in the Visual Editor

By default, the policy that you create using the visual editor allows the actions that you choose. To deny the chosen actions instead, choose **Switch to deny permissions**. Because requests are *denied by default*, we recommend as a security best practice that you allow permissions to only those actions and resources that a user needs. This is sometimes called "whitelisting." You should create a statement to deny permissions ("blacklisting") only if you want to override a permission separately that is allowed by another statement or policy. We recommend that you limit the number of deny permissions to a minimum because they can increase the difficulty of troubleshooting permissions. For more information about how IAM evaluates policy logic, see [IAM JSON Policy Evaluation Logic](#) (p. 458).

Note

By default, only the AWS account root user has access to all the resources in that account. So if you are not signed in as the root user, you must have permissions granted by a policy.

Specifying Multiple Services in the Visual Editor

When you use the visual editor to construct a policy, you can select only one service at a time. This is a best practice because the visual editor then allows you to choose from the actions for that one service. You then choose from the resources supported by that service and the selected actions. This makes it easier to create and troubleshoot your policy.

If you are familiar with the JSON syntax, you can also use a wildcard character (*) to manually specify multiple services. For example, type **Code*** to provide permissions for all services beginning with **Code**, such as **CodeBuild** and **CodeCommit**. However, you must then type the actions and resource ARNs to complete your policy. Additionally, when you save your policy, it might be [restructured](#) (p. 387) to include each service in a separate permission block.

Alternatively, to use JSON syntax (such as wildcards) for services, create, edit, and save your policy using the **JSON** tab.

Reducing the Size of Your Policy in the Visual Editor

When you use the visual editor to create or a policy, IAM creates a JSON document to store your policy. You can view this document by switching to the **JSON** tab. If this JSON document exceeds the size limit of a policy, the visual editor displays an error message and does not allow you to review and save your policy. To view the IAM limitation on the size of a managed policy, see [IAM Entity Character Limits](#) (p. 416).

To reduce the size of your policy in the visual editor, edit your policy or move permission blocks to another policy. The error message includes the number of characters that your policy document contains, and you can use this information to help you reduce the size of your policy.

Fixing Unrecognized Services, Actions, or Resource Types in the Visual Editor

When you create or edit a policy in the visual editor, you might see a warning that your policy includes an unrecognized service, action, or resource type.

Note

IAM reviews service names, actions, and resource types for services that support policy summaries. However, your policy summary might include a resource value or condition that does not exist. Always test your policies with the [policy simulator \(p. 322\)](#).

If your policy includes unrecognized services, actions or resource types, one of the following errors has occurred:

- **Preview service** – Services that are in preview do not support the visual editor. If you are participating in the preview, you can ignore the warning and continue, though you must manually type the actions and resource ARNs to complete your policy. Alternatively, you can choose the **JSON** tab to type or paste a JSON policy document.
- **Custom service** – Custom services do not support the visual editor. If you are using a custom service, you can ignore the warning and continue, though you must manually type the actions and resource ARNs to complete your policy. Alternatively, you can choose the **JSON** tab to type or paste a JSON policy document.
- **Service does not support the visual editor** – If your policy includes a generally available (GA) service that does not support the visual editor, you can ignore the warning and continue, though you must manually type the actions and resource ARNs to complete your policy. Alternatively, you can choose the **JSON** tab to type or paste a JSON policy document.

Generally available services are services that are released publicly and are not preview or custom services. If an unrecognized service is generally available and the name is spelled correctly, then the service does not support the visual editor. To learn how to request visual editor or policy summary support for a GA service, see [Service Does Not Support IAM Policy Summaries \(p. 391\)](#).

- **Action does not support the visual editor** – If your policy includes a supported service with an unsupported action, you can ignore the warning and continue, though you must manually type the resource ARNs to complete your policy. Alternatively, you can choose the **JSON** tab to type or paste a JSON policy document.

If your policy includes a supported service with an unsupported action, then the service does not fully support the visual editor. To learn how to request visual editor or policy summary support for a GA service, see [Service Does Not Support IAM Policy Summaries \(p. 391\)](#).

- **Resource type does not support the visual editor** – If your policy includes a supported action with an unsupported resource type, you can ignore the warning and continue. However, IAM cannot confirm that you have included resources for all of your selected actions, and you might see additional warnings.
- **Typo** – When you manually type a service, action, or resource in the visual editor, you can create a policy that includes a typo. As a best practice, use the visual editor by selecting from the list of services and actions, and then complete the resource section according to the prompts. However, if a service does not fully support the visual editor, you might have to manually type parts of your policy.

If you are certain that your policy contains none of the errors above, then your policy might include a typo. Check for misspelled service, action, and resource type names. For example, you might use `s2` instead of `s3` and `ListMyBuckets` instead of `ListAllMyBuckets`. Another common action typo is the inclusion of unnecessary text in ARNs, such as `arn:aws:s3: : :*`, or missing colons in actions, such as `AWSAuthRuntimeService.AuthenticatePassword`. You can evaluate a policy that might include typos by choosing **Review policy** to review the policy summary and confirm whether the policy provides the permissions you intended.

Troubleshoot Using Policy Summaries

You can diagnose and resolve issues related to policy summaries.

Missing Policy Summary

The IAM console includes *policy summary* tables that describe the access level, resources, and conditions that are allowed or denied for each service in a policy. Policies are summarized in three tables: the [policy summary \(p. 348\)](#), the [service summary \(p. 358\)](#), and the [action summary \(p. 362\)](#). The *policy summary* table includes a list of services and summaries of the permissions that are defined by the chosen policy. You can view the [policy summary \(p. 347\)](#) for any policies that are attached to a user on the **Users** page. You can view the policy summary for managed policies on the **Policies** page. If AWS is unable to render a summary for a policy, then you see the JSON policy document instead of the summary, and receive the following error:

A summary for this policy cannot be generated. You can still view or edit the JSON policy document.

If your policy does not include a summary, one of the following errors has occurred:

- **Unsupported policy element** – IAM does not support generating policy summaries for policies that include one of the following [policy elements \(p. 426\)](#):
 - Principal
 - NotPrincipal
 - NotResource
- **No policy permissions** – If a policy does not provide any effective permissions, then the policy summary cannot be generated. For example, if a policy includes a single statement with the element "NotAction": "*", then it grants access to all actions except "all actions" (*). This means it grants Deny or Allow access to nothing.

Note

You must be careful when using these policy elements such as `NotPrincipal`, `NotAction`, and `NotResource`. For information about using policy elements, see [IAM JSON Policy Elements Reference \(p. 426\)](#).

You can create a policy that does not provide effective permissions if you provide mismatched services and resources. This can occur if you specify actions in one service and resources from another service. In this case, the policy summary does appear. The only indication that there is a problem is that the resource column in the summary can include a resource from a different service. If this column includes a mismatched resource, then you should review your policy for errors. To better understand your policies, always test them with the [policy simulator \(p. 322\)](#).

Policy Summary Includes Unrecognized Services, Actions, or Resource Types

In the IAM console, if a [policy summary \(p. 347\)](#) includes a warning symbol (), then the policy might include an unrecognized service, action or resource type. To learn about warnings within a policy summary, see [Policy Summary \(List of Services\) \(p. 348\)](#).

Note

IAM reviews service names, actions, and resource types for services that support policy summaries. However, your policy summary might include a resource value or condition that does not exist. Always test your policies with the [policy simulator \(p. 322\)](#).

If your policy includes unrecognized services, actions or resource types, one of the following errors has occurred:

- **Preview service** – Services that are in preview do not support policy summaries.
- **Custom service** – Custom services do not support policy summaries.
- **Service does not support summaries** – If your policy includes a generally available (GA) service that does not support policy summaries, then the service is included in the **Unrecognized services** section of the policy summary table. Generally available services are services that are released publicly and are not preview or custom services. If an unrecognized service is generally available and the name is spelled correctly, then the service does not support IAM policy summaries. To learn how to request policy summary support for a GA service, see [Service Does Not Support IAM Policy Summaries \(p. 391\)](#).
- **Action does not support summaries** – If your policy includes a supported service with an unsupported action, then the action is included in the **Unrecognized actions** section of the service summary table. To learn about warnings within a service summary, see [Service Summary \(List of Actions\) \(p. 358\)](#).
- **Resource type does not support summaries** – If your policy includes a supported action with an unsupported resource type, then the resource is included in the **Unrecognized resource types** section of the service summary table. To learn about warnings within a service summary, see [Service Summary \(List of Actions\) \(p. 358\)](#).
- **Typo** – Because the policy validator in AWS checks only that the JSON is syntactically correct, you can create a policy that includes a typo. If you are certain that your policy contains none of the errors above, then your policy might include a typo. Check for misspelled service, action, and resource type names. For example, you might use s2 instead of s3 and ListMyBuckets instead of ListAllMyBuckets. Another common action typo is the inclusion of unnecessary text in ARNs, such as arn:aws:s3: : :*, or missing colons in actions, such as AWSAuthRuntimeService.AuthenticatePassword. You can evaluate a policy that might include typos by using the [policy simulator \(p. 322\)](#) to confirm whether the policy provides the permissions you intended.

Service Does Not Support IAM Policy Summaries

When a generally available (GA) service or action is not recognized by IAM policy summaries or the visual editor, it is possible that the service does not support these features. Generally available services are services that are released publicly and are not previewed or custom services. If an unrecognized service is generally available and the name is spelled correctly, then the service does not support these features. If your policy includes a supported service with an unsupported action, then the service does not fully support IAM policy summaries.

To request that a service add IAM policy summary or visual editor support

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Locate the policy that includes the unsupported service:
 - If the policy is a managed policy, choose **Policies** in the navigation pane. In the list of policies, choose the name of the policy that you want to view.
 - If the policy is an inline policy attached to the user, choose **Users** in the navigation pane. In the list of users, choose the name of the user whose policy you want to view. In the table of policies for the user, expand the header for the policy summary that you want to view.
3. In the left side on the AWS Management Console footer, choose **Feedback**. In the **Tell us about your experience:** box, type **I request that the <ServiceName> service add support for IAM policy summaries and the visual editor.** If you want more than one service to support summaries, type **I request that the <ServiceName1>, <ServiceName2>, and <ServiceName3> services add support for IAM policy summaries and the visual editor.**

To request that a service add IAM policy summary support for a missing action

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Locate the policy that includes the unsupported service:
 - If the policy is a managed policy, choose **Policies** in the navigation pane. In the list of policies, choose the name of the policy that you want to view.
 - If the policy is an inline policy attached to the user, choose **Users** in the navigation pane. In the list of users, choose the name of the user whose policy you want to view. In the table of policies for the user, choose the name of the policy that you want to view to expand the policy summary.
3. In the policy summary, choose the name of the service that includes an unsupported action.
4. In the left side on the AWS Management Console footer, choose **Feedback**. In the **Tell us about your experience:** box, type **I request that the <ServiceName> service add IAM policy summary and the visual editor support for the <ActionName> action.** If you want to report more than one unsupported action, type **I request that the <ServiceName> service add IAM policy summary and the visual editor support for the <ActionName1>, <ActionName2>, and <ActionName3> actions.**

To request that a different service includes missing actions, repeat the last three steps.

My Policy Does Not Grant the Expected Permissions

To assign permissions to a user, group, role, or resource, you create a *policy*, which is a document that defines permissions. The policy document includes the following elements:

- **Effect** – whether the policy allows or denies access
- **Action** – the list of actions that are allowed or denied by the policy
- **Resource** – the list of resources on which the actions can occur
- **Condition (Optional)** – the circumstances under which the policy grants permission

To learn about these and other policy elements, see [IAM JSON Policy Elements Reference \(p. 426\)](#).

To grant access, your policy must define an action with a supported resource. If your policy also includes a condition, that condition must include a [global condition key \(p. 476\)](#) or must apply to the action. To learn which resources are supported by an action, see the [AWS documentation](#) for your service. To learn which conditions are supported by an action, see [AWS Service Actions and Condition Context Keys for Use in IAM Policies \(p. 485\)](#).

To learn whether your policy defines an action, resource, or condition that does not grant permissions, you can view the [policy summary \(p. 348\)](#) for your policy using the IAM console at <https://console.aws.amazon.com/iam/>. You can use policy summaries to identify and correct problems in your policy.

There are several reasons why an element might not grant permissions despite being defined in the IAM policy:

- [An action is defined without an applicable resource \(p. 393\)](#)
- [A resource is defined without an applicable action \(p. 393\)](#)
- [A condition is defined without an applicable action \(p. 394\)](#)

To view examples of policy summaries that include warnings, see [the section called “Policy Summary \(List of Services\)” \(p. 348\)](#).

An Action Is Defined Without an Applicable Resource

The policy below defines all `ec2:Describe*` actions and defines a specific resource. None of the `ec2:Describe` actions are granted because none of these actions support resource-level permissions. Resource-level permissions mean that the action supports resources using ARNs in the policy's [Resource \(p. 436\)](#) element. If an action does not support resource-level permissions, then that statement in the policy must use a wildcard (*) in the Resource element. To learn which services support resource-level permissions, see [AWS Services That Work with IAM \(p. 417\)](#).

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": "ec2:Describe*",  
        "Resource": "arn:aws:ec2:us-east-2:ACCOUNT-ID:instance/*"  
    }]  
}
```

This policy does not provide any permissions, and the policy summary includes the following error:

This policy does not grant any permissions. To grant access, policies must have an action that has an applicable resource or condition.

To fix this policy, you must use * in the Resource element.

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": "ec2:Describe*",  
        "Resource": "*"  
    }]  
}
```

A Resource Is Defined Without an Applicable Action

The policy below defines an Amazon S3 bucket resource but does not include an S3 action that can be performed on that resource. This policy also grants full access to all Amazon CloudFront actions.

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": "cloudfront:*",  
        "Resource": [  
            "arn:aws:cloudfront:*,  
            "arn:aws:s3:::examplebucket"  
        ]  
    }]  
}
```

This policy provides permissions for all CloudFront actions. But because the policy defines the S3 `examplebucket` resource without defining any S3 actions, the policy summary includes the following warning:

This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition.

To fix this policy to provide S3 bucket permissions, you must define S3 actions that can be performed on a bucket resource.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudfront:*",
                "s3:CreateBucket",
                "s3>ListBucket*",
                "s3:PutBucket*",
                "s3:GetBucket*"
            ],
            "Resource": [
                "arn:aws:cloudfront:*",
                "arn:aws:s3:::examplebucket"
            ]
        }
    ]
}
```

Alternately, to fix this policy to provide only CloudFront permissions, remove the S3 resource.

A Condition Is Defined Without an Applicable Action

The policy below defines two Amazon S3 actions for all S3 resources, if the S3 prefix equals `custom` and the version ID equals `1234`. However, the `s3:VersionId` condition key is used for object version tagging and is not supported by the defined bucket actions. To learn which conditions are supported by an action, see [AWS Service Actions and Condition Context Keys for Use in IAM Policies \(p. 485\)](#) and follow the link to the service documentation for condition keys.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3>ListBucketVersions",
                "s3>ListBucket"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:prefix": [
                        "custom"
                    ],
                    "s3:VersionId": [
                        "1234"
                    ]
                }
            }
        }
    ]
}
```

This policy provides permissions for the `s3>ListBucket` action and the `s3>ListBucketVersions` action if the bucket name includes the `custom` prefix. But because the `s3:VersionId` condition is not supported by any of the defined actions, the policy summary includes the following error:

`This policy does not grant any permissions. To grant access, policies must have an action that has an applicable resource or condition.`

To fix this policy to use S3 object version tagging, you must define an S3 action that supports the `s3:VersionId` condition key.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3>ListBucketVersions",
                "s3>ListBucket",
                "s3>GetObjectVersion"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:prefix": [
                        "custom"
                    ],
                    "s3:VersionId": [
                        "1234"
                    ]
                }
            }
        }
    ]
}
```

This policy provides permissions for every action and condition in the policy. However, the policy still does not provide any permissions because there is no case where a single action matches both conditions. Instead, you must create two separate statements that each include only actions with the conditions to which they apply.

To fix this policy, create two statements. The first statement includes the actions that support the `s3:prefix` condition, and the second statement includes the actions that support the `s3:VersionId` condition.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3>ListBucketVersions",
                "s3>ListBucket"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:prefix": "custom"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "s3>GetObjectVersion",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:VersionId": "1234"
                }
            }
        }
    ]
}
```

Troubleshoot Policy Management

You can diagnose and resolve issues relating to policy management.

Attaching or Detaching a Policy in an IAM Account

Some AWS managed policies are linked to a service. These policies are used only with a [service-linked role \(p. 136\)](#) for that service. In the IAM console, when you view the **Summary** page for a policy, the page includes a banner to indicate that the policy is linked to a service. You cannot attach this policy to a user, group, or role within IAM. When you create a service-linked role for the service, this policy is automatically attached to your new role. Because the policy is required, you cannot detach the policy from the service-linked role.

Troubleshoot JSON Policy Documents

You can diagnose and resolve issues relating to JSON policy documents.

More Than One JSON Policy Object

An IAM policy 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": "Allow",  
        "Action": "s3:*",  
        "Resource": "arn:aws:s3:::my-bucket/*"  
    }  
}
```

You can, however, meet the intention of the previous 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 (called out in bold):

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "ec2:Describe*",  
            "Resource": "*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": "s3:*",  
            "Resource": "arn:aws:s3:::my-bucket/*"  
        }  
    ]  
}
```

```
}
```

More Than One JSON Statement Element

This error might at first appear to be a variation on the previous section. However, syntactically it is a different type of error. The following example has 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 IAM policy 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 (called out in *red*):

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "ec2:Describe*",
        "Resource": "*"
    },
    "Statement": {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::my-bucket/*"
    }
}
```

A value object can be an array of multiple value objects. To solve this problem, combine the two `Statement` elements into one element with an object array, as shown in the following example (called out in **bold**):

```
{
    "Version": "2012-10-17",
    "Statement": [   {
        "Effect": "Allow",
        "Action": "ec2:Describe*",
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::my-bucket/*"
    }
]
```

The value of the `Statement` element is an object array. The array in this example consists of two objects, each of which is by itself a correct value for a `Statement` element. Each object in the array is separated by commas.

More Than One Effect, Action, or Resource Element in a JSON Statement Element

On the value side of the `Statement` name/value pair, the object must consist of only one `Effect` element, one `Action` element, and one `Resource` element. The following policy is incorrect because it has two `Effect` elements in the value object of the `Statement`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Effect": "Allow",
            "Action": "ec2:*",
            "Resource": "*"
        }
    ]
}
```

Note

The policy engine does not allow such errors in new or edited policies. However, the policy engine continues to permit policies that were saved before the engine was updated. The behavior of existing policies with the error is as follows:

- Multiple `Effect` elements: only the last `Effect` element is observed. The others are ignored.
- Multiple `Action` elements: all `Action` elements are combined internally and treated as if they were a single list.
- Multiple `Resource` elements: all `Resource` elements are combined internally and treated as if they were a single list.

The policy engine does not allow you to save any policy with syntax errors. You must correct the errors in the policy before you can save it. The [Policy Validator \(p. 321\)](#) tool can help you to find all older policies with errors and can recommend corrections for them.

In each case, the solution is to remove the incorrect extra element. For `Effect` elements, this is straightforward: if you want the previous example to *deny* permissions to Amazon EC2 instances, then you must remove the line `"Effect": "Allow"`, from the policy, as follows:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "ec2:*",
            "Resource": "*"
        }
    ]
}
```

However, if the duplicate element is `Action` or `Resource`, then the resolution can be more complicated. You might have multiple actions that you want to allow (or deny) permission to, or you might want to control access to multiple resources. For example, the following example is incorrect because it has multiple `Resource` elements (called out in red):

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::my-bucket",
            "Resource": "arn:aws:s3:::my-bucket/*"
        }
    ]
}
```

Each of the required elements in a `Statement` element's value object can be present only once. The solution is to place each value in an array. The following example illustrates this by making the two separate resource elements into one `Resource` element with an array as the value object (called out in bold):

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "s3:*",  
            "Resource": [  
                "arn:aws:s3:::my-bucket",  
                "arn:aws:s3:::my-bucket/*"  
            ]  
        }  
    ]  
}
```

Missing JSON Version Element

A `Version` policy element is different from a policy version. The `Version` policy element is used within a policy and defines the version of the policy language. A policy version, on the other hand, is created when you make changes to a customer managed policy in IAM. The changed policy doesn't overwrite the existing policy. Instead, IAM creates a new version of the managed policy. To learn more about the `Version` policy element see [IAM JSON Policy Elements: Version \(p. 426\)](#). To learn more about policy versions, see [the section called "Versioning IAM Policies" \(p. 335\)](#).

As AWS features evolve, new capabilities are added to IAM policies to support those features. Sometimes, an update to the policy syntax includes a new version number. If you use newer features of the policy grammar in your policy, then you must tell the policy parsing engine which version you are using. The default policy version is "2008-10-17." If you want to use any policy feature that was introduced later, then you must specify the version number that supports the feature you want. We recommend that you *always* include the latest policy syntax version number, which is currently "`Version": "2012-10-17"`. For example, the following policy is incorrect because it uses a policy variable `#{...}` in the ARN for a resource without specifying a policy syntax version that supports policy variables (called out in *red*):

```
{  
    "Statement": [  
        {  
            "Action": "iam:*AccessKey*",  
            "Effect": "Allow",  
            "Resource": "arn:aws:iam::123456789012:user/${aws:username}"  
        }  
    ]  
}
```

Adding a `Version` element at the top of the policy with the value `2012-10-17`, the first IAM API version that supports policy variables, solves this problem (called out in **bold**):

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": "iam:*AccessKey*",  
            "Effect": "Allow",  
            "Resource": "arn:aws:iam::123456789012:user/${aws:username}"  
        }  
    ]  
}
```

Troubleshooting IAM Roles

Use the information here to help you diagnose and fix common issues that you might encounter when working with IAM roles.

Topics

- [I Can't Assume a Role \(p. 400\)](#)
- [A New Role Appeared in My AWS Account \(p. 400\)](#)
- [I Can't Edit or Delete a Role in My AWS Account \(p. 401\)](#)

I Can't Assume a Role

- Verify that your IAM policy grants you permission to call `sts:AssumeRole` for the role that you want to assume. The `Action` element of your IAM policy must allow you to call the `AssumeRole` action, and the `Resource` element of your IAM policy must specify the role that you want to assume. For example, the `Resource` element can specify a role by its Amazon Resource Name (ARN) or by using a wildcard (*). For example, at least one policy applicable to you must grant permissions similar to the following:

```
"Effect": "Allow",
"Action": "sts:AssumeRole",
"Resource": "arn:aws:iam::account_id_number:role/role-name-you-want-to-assume"
```

- Verify that you meet all the conditions that are specified in the role's trust policy. A `Condition` can specify an expiration date, an external ID, or that a request must come only from specific IP addresses. In the following example, if the current date is any time after the specified date, then the policy never matches and cannot grant you the permission to assume the role.

```
"Effect": "Allow",
"Action": "sts:AssumeRole",
"Resource": "arn:aws:iam::account_id_number:role/role-name-you-want-to-assume"
"Condition": {
    "DateLessThan" : {
        "aws:CurrentTime" : "2016-05-01T12:00:00Z"
    }
}
```

- Verify that the AWS account that you are calling `AssumeRole` from is a trusted entity for the role that you are assuming. Trusted entities are defined as a `Principal` in a role's trust policy. The following example is a trust policy attached to the role you want to assume. In this example, the account ID with the IAM user you signed-in with must be 123456789012. If your account number is not listed in the `Principal` element of the role's trust policy, then you cannot assume the role, no matter what permissions are granted to you in access policies. Note that the example policy limits permissions to actions that occur between July 1, 2017 and December 31, 2017 (UTC), inclusive. If you log in before or after those dates, then the policy does not match, and you cannot assume the role.

```
"Effect": "Allow",
"Principal": { "AWS": "arn:aws:iam::123456789012:root" },
"Action": "sts:AssumeRole",
"Condition": {
    "DateGreaterThanOrEqual": { "aws:CurrentTime": "2017-07-01T00:00:00Z" },
    "DateLessThan": { "aws:CurrentTime": "2017-12-31T23:59:59Z" }
}
```

A New Role Appeared in My AWS Account

Some AWS services require that you use a unique type of service role that is linked directly to the service. This [service-linked role \(p. 136\)](#) is predefined by the service and includes all the permissions that the service requires. This makes setting up a service easier because you don't have to manually add the

necessary permissions. For general information about service-linked roles, see [Using Service-Linked Roles \(p. 176\)](#).

You might already be using a service when it begins supporting service-linked roles. If so, you might receive an email telling you about a new role in your account. This role includes all the permissions that the service needs to perform actions on your behalf. You don't need to take any action to support this role. However, you should not delete the role from your account. Doing so could remove permissions that the service needs to access AWS resources. You can view the service-linked roles in your account by going to the **IAM Roles** page of the IAM console. Service-linked roles appear with **(Service-linked role)** in the **Trusted entities** column of the table.

For information about which services support service-linked roles, see [AWS Services That Work with IAM \(p. 417\)](#) and look for the services that have **Yes** in the **Service-Linked Role** column. For information about using the service-linked role for a service, choose the **Yes** link.

I Can't Edit or Delete a Role in My AWS Account

You cannot delete or edit the permissions for a [service-linked role \(p. 136\)](#) in IAM. These roles include predefined trusts and permissions that are required by the service in order to perform actions on your behalf. You can use the IAM console, AWS CLI, or API to edit only the description of a service-linked role. You can view the service-linked roles in your account by going to the **IAM Roles** page in the console. Service-linked roles appear with **(Service-linked role)** in the **Trusted entities** column of the table. A banner on the role's **Summary** page also indicates that the role is a service-linked role. You can manage and delete these roles only through the linked service, if that service supports the action. Be careful when modifying or deleting a service-linked role because doing so could remove permissions that the service needs to access AWS resources.

For information about which services support service-linked roles, see [AWS Services That Work with IAM \(p. 417\)](#) and look for the services that have **Yes** in the **Service-Linked Role** column.

Troubleshooting Amazon EC2 and IAM

Use the information here to help you troubleshoot and fix access denied or other issues that you might encounter when working with Amazon EC2 and IAM.

Topics

- [When attempting to launch an instance, I don't see the role I expected to see in the Amazon EC2 console IAM role list. \(p. 401\)](#)
- [The credentials on my instance are for the wrong role. \(p. 402\)](#)
- [When I attempt to call the AddRoleToInstanceProfile, I get an AccessDenied error. \(p. 402\)](#)
- [Amazon EC2: When I attempt to launch an instance with a role, I get an AccessDenied error. \(p. 402\)](#)
- [I can't access the temporary security credentials on my EC2 instance. \(p. 403\)](#)
- [What do the errors from the info document in the IAM subtree mean? \(p. 403\)](#)

When attempting to launch an instance, I don't see the role I expected to see in the Amazon EC2 console IAM role list.

Check the following:

- If you are signed in as an IAM user, verify that you have permission to call `ListInstanceProfiles`. For information about the permissions necessary to work with roles, see "Permissions Required for Using Roles with Amazon EC2" in [Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances \(p. 215\)](#). For information about adding permissions to a user, see [Managing IAM Policies \(p. 316\)](#).

If you cannot modify your own permissions, you must contact an administrator who can work with IAM in order to update your permissions.

- If you created a role by using the IAM CLI or API, verify that you created an instance profile and added the role to that instance profile. Also, if you name your role and instance profile differently, you won't see the correct role name in the list of IAM roles in the Amazon EC2 console. The **IAM Role** list in the Amazon EC2 console lists the names of instance profiles, not the names of roles. You will have to select the name of the instance profile that contains the role you want. For details about instance profiles, see [Using Instance Profiles \(p. 218\)](#).

Note

If you use the IAM console to create roles, you don't need to work with instance profiles. For each role that you create in the IAM console, an instance profile is created with the same name as the role, and the role is automatically added to that instance profile. An instance profile can contain only one IAM role, and that limit cannot be increased.

The credentials on my instance are for the wrong role.

If the role in the instance profile was replaced recently, your application will need to wait for the next automatically scheduled credential rotation before credentials for your role become available.

When I attempt to call the `AddRoleToInstanceProfile`, I get an `AccessDenied` error.

If you are making requests as an IAM user, verify that you have the following permissions:

- `iam:AddRoleToInstanceProfile` with the resource matching the instance profile ARN (for example, `arn:aws:iam::999999999999:instance-profile/ExampleInstanceProfile`).

For more information about the permissions necessary to work with roles, see "How Do I Get Started?" in [Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances \(p. 215\)](#). For information about adding permissions to a user, see [Managing IAM Policies \(p. 316\)](#).

Amazon EC2: When I attempt to launch an instance with a role, I get an `AccessDenied` error.

Check the following:

- Launch an instance without an instance profile. This will help ensure that the problem is limited to IAM roles for Amazon EC2 instances.
- If you are making requests as an IAM user, verify that you have the following permissions:
 - `ec2:RunInstances` with a wildcard resource ("*")
 - `iam:PassRole` with the resource matching the role ARN (for example, `arn:aws:iam::999999999999:role/ExampleRoleName`)

- Call the IAM `GetInstanceProfile` action to ensure that you are using a valid instance profile name or a valid instance profile ARN. For more information, see [Using IAM roles with Amazon EC2 instances](#).
- Call the IAM `GetInstanceProfile` action to ensure that the instance profile has a role. Empty instance profiles will fail with an `AccessDenied` error. For more information about creating a role, see [Creating IAM Roles \(p. 184\)](#).

For more information about the permissions necessary to work with roles, see "How Do I Get Started?" in [Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances \(p. 215\)](#). For information about adding permissions to a user, see [Managing IAM Policies \(p. 316\)](#).

I can't access the temporary security credentials on my EC2 instance.

Check the following:

- Can you access another part of the instance metadata service (IMDS)? If not, check that you have no firewall rules blocking access to requests to the IMDS.

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/hostname; echo
```

- Does the `iam` subtree of the IMDS exist? If not, verify that your instance has an IAM instance profile associated with it by calling `ec2:DescribeInstances`.

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/iam; echo
```

- Check the `info` document in the `iam` subtree for an error. If you have an error, see [What do the errors from the info document in the IAM subtree mean? \(p. 403\)](#) for more information.

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/iam/info; echo
```

What do the errors from the `info` document in the IAM subtree mean?

The `iam/info` document indicates
"Code": "InstanceProfileNotFound".

Your IAM instance profile has been deleted and Amazon EC2 can no longer provide credentials to your instance. You will need to terminate your instances and restart with a valid instance profile.

If an instance profile with that name exists, check that the instance profile wasn't deleted and another was created with the same name.

To verify the status of the instance profile:

1. Call the IAM `GetInstanceProfile` action to get the `InstanceProfileId`.
2. Call the Amazon EC2 `DescribeInstances` action to get the `IamInstanceProfileId` for the instance.

3. Verify that the `InstanceProfileId` from the IAM action matches the `IamInstanceProfileId` from the Amazon EC2 action.

If the IDs are different, then the instance profile attached to your instances is no longer valid. You will need to terminate your instances and restart with a valid instance profile.

The `iam/info` document indicates a success but indicates "Message": "Instance Profile does not contain a role..."

The role has been removed from the instance profile by the IAM `RemoveRoleFromInstanceProfile` action. You can use the IAM `AddRoleToInstanceProfile` action to attach a role to the instance profile. Your application will need to wait until the next scheduled refresh to access the credentials for the role.

The `iam/security-credentials/[role-name]` document indicates "Code": "AssumeRoleUnauthorizedAccess".

Amazon EC2 does not have permission to assume the role. Permission to assume the role is controlled by the trust policy attached to the role, like the example that follows. Use the IAM `UpdateAssumeRolePolicy` API to update the trust policy.

```
{"Version": "2012-10-17", "Statement": [{"Effect": "Allow", "Principal": {"Service": ["ec2.amazonaws.com"]}, "Action": ["sts:AssumeRole"]}]}
```

Your application will need to wait until the next automatically scheduled refresh to access the credentials for the role.

Troubleshooting Amazon S3 and IAM

Use the information here to help you troubleshoot and fix issues that you might encounter when working with Amazon S3 and IAM.

How do I grant anonymous access to an Amazon S3 bucket?

You use an Amazon S3 bucket policy that specifies a wildcard (*) in the `principal` element, which means anyone can access the bucket. With anonymous access, anyone (including users without an AWS account) will be able to access the bucket. For a sample policy, see [Example Cases for Amazon S3 Bucket Policies](#) in the *Amazon Simple Storage Service Developer Guide*.

I'm signed in as an AWS account root user, why can't I access an Amazon S3 bucket under my account?

In some cases, you might have an IAM user with full access to IAM and Amazon S3. If the IAM user assigns a bucket policy to an Amazon S3 bucket and doesn't specify the AWS account root user as a principal, the root user is denied access to that bucket. However, as the root user, you can still access the bucket by modifying the bucket policy to allow root user access using the Amazon S3 console or the AWS CLI.

Troubleshooting SAML 2.0 Federation with AWS

Use the information here to help you diagnose and fix issues that you might encounter when working with SAML 2.0 and federation with IAM.

Topics

- [Error: Your request included an invalid SAML response. To logout, click here. \(p. 405\)](#)
- [Error: RoleSessionName is required in AuthnResponse \(Service: AWSSecurityTokenService; Status Code: 400; Error Code: InvalidIdentityToken\) \(p. 405\)](#)
- [Error: Not authorized to perform sts:AssumeRoleWithSAML \(Service: AWSSecurityTokenService; Status Code: 403; Error Code: AccessDenied\) \(p. 406\)](#)
- [Error: RoleSessionName in AuthnResponse must match \[a-zA-Z_0-9+=,.@-\]{2,64} \(Service: AWSSecurityTokenService; Status Code: 400; Error Code: InvalidIdentityToken\) \(p. 406\)](#)
- [Error: Response signature invalid \(Service: AWSSecurityTokenService; Status Code: 400; Error Code: InvalidIdentityToken\) \(p. 406\)](#)
- [Error: Failed to assume role: Issuer not present in specified provider \(Service: AWSOpenIdDiscoveryService; Status Code: 400; Error Code: AuthSamlInvalidSamlResponseException\) \(p. 407\)](#)
- [Error: Could not parse metadata. \(p. 407\)](#)
- [How to View a SAML Response in Your Browser for Troubleshooting \(p. 407\)](#)

Error: Your request included an invalid SAML response. To logout, click here.

This error can occur when the SAML response from the identity provider does not include an attribute with the Name set to `https://aws.amazon.com/SAML/Attributes/Role`. The attribute must contain one or more `AttributeValue` elements, each containing a comma-separated pair of strings:

- The ARN of a role that the user can be mapped to
- The ARN of the SAML provider

For more information, see [Configuring SAML Assertions for the Authentication Response \(p. 163\)](#). To view the SAML response in your browser, follow the steps listed in [How to View a SAML Response in Your Browser for Troubleshooting \(p. 407\)](#).

Error: RoleSessionName is required in AuthnResponse (Service: AWSSecurityTokenService; Status Code: 400; Error Code: InvalidIdentityToken)

This error can occur when the SAML response from the identity provider does not include an attribute with the Name set to `https://aws.amazon.com/SAML/Attributes/RoleSessionName`. The attribute value is an identifier for the user and is typically a user ID or an email address.

For more information, see [Configuring SAML Assertions for the Authentication Response \(p. 163\)](#). To view the SAML response in your browser, follow the steps listed in [How to View a SAML Response in Your Browser for Troubleshooting \(p. 407\)](#).

Error: Not authorized to perform sts:AssumeRoleWithSAML (Service: AWSSecurityTokenService; Status Code: 403; Error Code: AccessDenied)

This error can occur if the IAM role specified in the SAML response is misspelled or does not exist. Correct the name of the role in the SAML service provider configuration.

This error can also occur if the federated users do not have permissions to assume the role. The role must have a trust policy that specifies the ARN of the IAM SAML identity provider as the Principal. The role also contains conditions that control which users can assume the role. Ensure that your users meet the requirements of the conditions.

This error can also occur if the SAML response does not include a Subject containing a NameID.

For more information see [Establish Permissions in AWS for Federated Users](#) and [Configuring SAML Assertions for the Authentication Response \(p. 163\)](#). To view the SAML response in your browser, follow the steps listed in [How to View a SAML Response in Your Browser for Troubleshooting \(p. 407\)](#).

Error: RoleSessionName in AuthnResponse must match [a-zA-Z_0-9+=,.@-]{2,64} (Service: AWSSecurityTokenService; Status Code: 400; Error Code: InvalidIdentityToken)

This error can occur if the RoleSessionName attribute value is too long or contains invalid characters. The maximum valid length is 64 characters.

For more information, see [Configuring SAML Assertions for the Authentication Response \(p. 163\)](#). To view the SAML response in your browser, follow the steps listed in [How to View a SAML Response in Your Browser for Troubleshooting \(p. 407\)](#).

Error: Response signature invalid (Service: AWSSecurityTokenService; Status Code: 400; Error Code: InvalidIdentityToken)

This error can occur when federation metadata of the identity provider does not match the metadata of the IAM identity provider. For example, the metadata file for the identity service provider might have changed to update an expired certificate. Download the updated SAML metadata file from your identity service provider. Then update it in the AWS identity provider entity that you define in IAM with the `aws iam update-saml-provider` cross-platform CLI command or the `Update-IAMSAMLProvider` PowerShell cmdlet.

Error: Failed to assume role: Issuer not present in specified provider (Service: AWSOpenIdDiscoveryService; Status Code: 400; Error Code: AuthSamlInvalidSamlResponseException)

This error can occur if the issuer in the SAML response does not match the issuer declared in the federation metadata file uploaded to AWS when you created the identity provider in IAM.

Error: Could not parse metadata.

This error can occur if your metadata file is not formatted properly.

When you [create or manage a SAML identity provider \(p. 159\)](#) in the AWS Management Console, you must retrieve the SAML metadata document from your identity provider. This metadata file includes the issuer's name, expiration information, and keys that can be used to validate the SAML authentication response (assertions) that are received from the IdP. The metadata file must be encoded in UTF-8 format without a byte order mark (BOM). Also, the x.509 certificate that is included as part of the SAML metadata document must use a key size of at least 1024 bits. If the key size is smaller, the IdP creation fails with an "Unable to parse metadata" error. To remove the BOM, you can encode the file as UTF-8 using a text editing tool, such as Notepad++.

How to View a SAML Response in Your Browser for Troubleshooting

The following procedures describe how to view the SAML response from your service provider from in your browser when troubleshooting a SAML 2.0-related issue.

For all browsers, go to the page where you can reproduce the issue. Then follow the steps for the appropriate browser:

Topics

- [Google Chrome \(p. 407\)](#)
- [Mozilla Firefox \(p. 408\)](#)
- [Apple Safari \(p. 408\)](#)
- [Microsoft Internet Explorer \(p. 408\)](#)
- [What to do with the Base64-encoded SAML response \(p. 408\)](#)

Google Chrome

To view a SAML response in Chrome

These steps were tested using version 54.0.2840.87m. If you use another version, you might need to adapt the steps accordingly.

1. Press **F12** to start the developer console.
2. Select the **Network** tab, and then select **Preserve log**.
3. Reproduce the issue.
4. Look for a **SAML Post** in the developer console pane. Select that row, and then view the **Headers** tab at the bottom. Look for the **SAMLResponse** attribute that contains the encoded request.

Mozilla Firefox

To view a SAML response in Firefox

This procedure was tested on version 37.0.2 of Mozilla Firefox. If you use another version, you might need to adapt the steps accordingly.

1. Press **F12** to start the developer console.
2. In the upper right of the developer tools window, click options (the small gear icon). Under **Common Preferences** select **Enable persistent logs**.
3. Select the **Network** tab.
4. Reproduce the issue.
5. Look for a **POST SAML** in the table. Select that row. In the **Form Data** window on the right, select the **Params** tab and find the **SAMLResponse** element.

Apple Safari

To view a SAML response in Safari

These steps were tested using version 8.0.6 (10600.6.3). If you use another version, you might need to adapt the steps accordingly.

1. Enable Web Inspector in Safari. Open the **Preferences** window, select the **Advanced** tab, and then select **Show Develop menu in the menu bar**.
2. Now you can open Web Inspector. Click **Develop**, then select **Show Web Inspector**.
3. Select the **Resources** tab.
4. Reproduce the issue.
5. Look for a **saml-signin.aws.amazon.com** request.
6. Scroll down to find Request Data with the name **SAMLResponse**. The associated value is the Base64-encoded response.

Microsoft Internet Explorer

To view a SAML response in Internet Explorer

The best way analyze network traffic in Internet Explorer is through the use of a third-party tool.

- Follow the steps at <http://social.technet.microsoft.com/wiki/contents/articles/3286.ad-fs-2-0-how-to-use-fiddler-web-debugger-to-analyze-a-ws-federation-passive-sign-in.aspx> to download and install Fiddler and capture the data.

What to do with the Base64-encoded SAML response

Once you find the Base64-encoded SAML response element in your browser, copy it and use your favorite Base-64 decoding tool to extract the XML tagged response.

Security Tip

Because the SAML response data that you are viewing might contain sensitive security data, we recommend that you do not use an *online* base64 decoder. Instead use a tool installed on your local computer that does not send your SAML data over the network.

Built-in option for Windows systems (PowerShell):

```
PS C:\> [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String("base64encodedtext"))
```

Built-in option for MacOS and Linux systems:

```
$ echo "base64encodedtext" | base64 --decode
```

Reference Information for AWS Identity and Access Management

Use the topics in this section to find detailed reference material for various aspects of IAM and AWS STS.

Topics

- [IAM Identifiers \(p. 410\)](#)
- [Limitations on IAM Entities and Objects \(p. 414\)](#)
- [AWS Services That Work with IAM \(p. 417\)](#)
- [IAM JSON Policy Reference \(p. 425\)](#)

IAM Identifiers

IAM uses a few different identifiers for users, groups, roles, policies, and server certificates. This section describes the identifiers and when you use each.

Topics

- [Friendly Names and Paths \(p. 410\)](#)
- [IAM ARNs \(p. 410\)](#)
- [Unique IDs \(p. 413\)](#)

Friendly Names and Paths

When you create a user, a role, a group, or a policy, or when you upload a server certificate, you give it a friendly name, such as Bob, TestApp1, Developers, ManageCredentialsPermissions, or ProdServerCert.

If you are using the IAM API or AWS Command Line Interface (AWS CLI) to create IAM entities, you can also give the entity an optional path. You can use a single path, or nest multiple paths as if they were a folder structure. For example, you could use the nested path /division_abc/subdivision_xyz/product_1234/engineering/ to match your company's organizational structure. You could then create a policy to allow all users in that path to access the policy simulator API. To view this policy, see [IAM: Access the Policy Simulator API Based on User Path \(p. 307\)](#). For additional examples of how you might use paths, see [IAM ARNs \(p. 410\)](#).

Just because you give a user and group the same path doesn't automatically put that user in that group. For example, you might create a Developers group and specify its path as /division_abc/subdivision_xyz/product_1234/engineering/. Just because you create a user named Bob and give him that same path doesn't automatically put Bob in the Developers group. IAM doesn't enforce any boundaries between users or groups based on their paths. Users with different paths can use the same resources (assuming they've been granted permission to those resources). For information about limitations on names, see [Limitations on IAM Entities and Objects \(p. 414\)](#).

IAM ARNs

Most resources have a friendly name (for example, a user named Bob or a group named Developers). However, the permission policy language requires you to specify the resource or resources using the following *Amazon Resource Name (ARN)* format.

```
arn:partition:service:region:account:resource
```

Where:

- **partition** identifies the partition that the resource is in. For standard AWS regions, the partition is `aws`. If you have resources in other partitions, the partition is `aws-partitionname`. For example, the partition for resources in the China (Beijing) region is `aws-cn`.
- **service** identifies the AWS product. For IAM resources, this is always `iam`.
- **region** is the region the resource resides in. For IAM resources, this is always left blank.
- **account** is the AWS account ID with no hyphens (for example, 123456789012).
- **resource** is the portion that identifies the specific resource by name.

You can use ARNs in IAM for users (IAM and federated), groups, roles, policies, instance profiles, virtual MFA devices, and [server certificates \(p. 125\)](#). The following table shows the ARN format for each and an example. The region portion of the ARN is blank because IAM resources are global.

Note

Many of the following examples include paths in the resource part of the ARN. Paths cannot be created or manipulated in the AWS Management Console. To use paths you must work with the resource by using the AWS API, the AWS CLI, or the Tools for Windows PowerShell.

The following examples show ARNs for different types of IAM resources.

- *The AWS account - the account itself:*

```
arn:aws:iam::123456789012:root
```

- *An IAM user in the account:*

```
arn:aws:iam::123456789012:user/Bob
```

- *Another user with a path reflecting an organization chart:*

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/Bob
```

- *An IAM group:*

```
arn:aws:iam::123456789012:group/Developers
```

- *An IAM group with a path:*

```
arn:aws:iam::123456789012:group/division_abc/subdivision_xyz/product_A/  
Developers
```

- *An IAM role:*

```
arn:aws:iam::123456789012:role/S3Access
```

- *A managed policy:*

```
arn:aws:iam::123456789012:policy/ManageCredentialsPermissions
```

- *An instance profile that can be associated with an EC2 instance:*

```
arn:aws:iam::123456789012:instance-profile/Webserver
```

- *A federated user identified in IAM as "Bob":*

```
arn:aws:sts::123456789012:federated-user/Bob
```

- *The active session of someone assuming the role of "Accounting-Role", with a role session name of "Mary":*

```
arn:aws:sts::123456789012:assumed-role/Accounting-Role/Mary
```

- *The multi-factor authentication device assigned to the user named Bob:*

```
arn:aws:iam::123456789012:mfa/Bob
```

- *A server certificate*

```
arn:aws:iam::123456789012:server-certificate/ProdServerCert
```

- *A server certificate with a path that reflects an organization chart:*

```
arn:aws:iam::123456789012:server-certificate/division_abc/subdivision_xyz/  
ProdServerCert
```

- *Identity providers (SAML and OIDC):*

```
arn:aws:iam::123456789012:saml-provider/ADFSProvider
```

```
arn:aws:iam::123456789012:oidc-provider/GoogleProvider
```

The following example shows a policy you could assign to Bob to allow him to manage his own access keys. Notice that the resource is the IAM user Bob.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": ["iam:*AccessKey*"],  
            "Resource": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:user/division_abc/subdivision_xyz/  
Bob"  
        }  
    ]  
}
```

Note

When you use ARNs to identify resources in an IAM policy, you can include *policy variables* that let you include placeholders for run-time information (such as the user's name) as part of the ARN. For more information, see [IAM Policy Elements: Variables \(p. 451\)](#)

You can use wildcards in the *resource* portion of the ARN to specify multiple users or groups or policies. For example, to specify all users working on product_1234, you would use:

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/product_1234/*
```

Let's say you have users whose names start with the string app_. You could refer to them all with the following ARN.

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/product_1234/app_*
```

To specify all users, groups, or policies in your AWS account, use a wildcard after the `user/`, `group/`, or `policy` part of the ARN, respectively.

```
arn:aws:iam::123456789012:user/*  
arn:aws:iam::123456789012:group/*  
arn:aws:iam::123456789012:policy/*
```

Don't use a wildcard in the `user/`, `group/`, or `policy` part of the ARN. In other words, the following is not allowed:

```
arn:aws:iam::123456789012:u*
```

Example Use of Paths and ARNs for a Project-Based Group

Paths cannot be created or manipulated in the AWS Management Console. To use paths you must work with the resource by using the AWS API, the AWS CLI, or the Tools for Windows PowerShell.

In this example, Jules in the Marketing_Admin group creates a project-based group within the /marketing/ path, and assigns users from different parts of the company to the group. This example illustrates that a user's path isn't related to the groups the user is in.

The marketing group has a new product they'll be launching, so Jules creates a new group in the /marketing/ path called Widget_Launch. Jules then assigns the following policy to the group, which gives the group access to objects in the part of the example_bucket designated to this particular launch.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "s3:*",  
            "Resource": "arn:aws:s3:::example_bucket/marketing/newproductlaunch/widget/*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": "s3>ListBucket*",  
            "Resource": "arn:aws:s3:::example_bucket",  
            "Condition": {"StringLike": {"s3:prefix": "marketing/newproductlaunch/widget/*"}}  
        }  
    ]  
}
```

Jules then assigns the users who are working on this launch to the group. This includes Patricia and Eli from the /marketing/ path. It also includes Chris and Chloe from the /sales/ path, and Aline and Jim from the /legal/ path.

Unique IDs

When IAM creates a user, group, role, policy, instance profile, or server certificate, it assigns to each entity a unique ID that looks like the following example:

AIDAJQABLZS4A3QDU576Q

For the most part, you use friendly names and [ARNs](#) when you work with IAM entities, so you don't need to know the unique ID for a specific entity. However, the unique ID can sometimes be useful when it isn't practical to use friendly names.

One example pertains to reusing friendly names in your AWS account. Within your account, a friendly name for a user, group, or policy must be unique. For example, you might create an IAM user named David. Your company uses Amazon S3 and has a bucket with folders for each employee; the bucket has a resource-based policy (a bucket policy) that lets users access only their own folders in the bucket. Suppose that the employee named David leaves your company and you delete the corresponding IAM user. But later another employee named David starts and you create a new IAM user named David. If the bucket policy specifies the IAM user named David, the policy could end up granting the new David access to information in the Amazon S3 bucket that was left by the former David.

However, every IAM user has a unique ID, even if you create a new IAM user that reuses a friendly name that you deleted before. In the example, the old IAM user David and the new IAM user David have different unique IDs. If you create resource policies for Amazon S3 buckets that grant access by unique ID and not just by user name, it reduces the chance that you could inadvertently grant access to information that an employee should not have.

Another example where user IDs can be useful is if you maintain your own database (or other store) of IAM user information. The unique ID can provide a unique identifier for each IAM user you create, even if over time you have IAM users that reuse a name, as in the previous example.

Getting the Unique ID

The unique ID for an IAM entity is not available in the IAM console. To get the unique ID, you can use the following AWS CLI commands or IAM API calls.

AWS CLI:

- [get-group](#)
- [get-role](#)
- [get-user](#)
- [get-policy](#)
- [get-instance-profile](#)
- [get-server-certificate](#)

IAM API:

- [GetGroup](#)
- [GetRole](#)
- [GetUser](#)
- [GetPolicy](#)
- [GetInstanceProfile](#)
- [GetServerCertificate](#)

Limitations on IAM Entities and Objects

Entities and objects in IAM have size limitations. IAM limits how you name an entity, the number of entities you can create, and the number of characters you can use in an entity.

Note

To get account-level information about entity usage and quotas, use the [GetAccountSummary](#) API action or the [get-account-summary](#) AWS CLI command.

IAM Entity Name Limits

The following are restrictions on IAM names:

- Policy documents can contain only the following Unicode characters: horizontal tab (U+0009), linefeed (U+000A), carriage return (U+000D), and characters in the range U+0020 to U+00FF.
- Names of users, groups, roles, policies, instance profiles, and server certificates must be alphanumeric, including the following common characters: plus (+), equal (=), comma (,), period (.), at (@), underscore (_), and hyphen (-).
- Names of users, groups, and roles must be unique within the account. They are not distinguished by case, for example, you cannot create groups named both "ADMINS" and "admins".
- Path names must begin and end with a forward slash (/).
- Policy names for [inline policies \(p. 279\)](#) must be unique to the user, group, or role they are embedded in, and can contain any Basic Latin (ASCII) characters minus the following reserved characters:

backward slash (\), forward slash (/), asterisk (*), question mark (?), and white space. These characters are reserved according to [RFC 3986](#).

- User passwords (login profiles) can contain any Basic Latin (ASCII) characters.
- AWS account ID aliases must be unique across AWS products, and must be alphanumeric following DNS naming conventions. An alias must be lowercase, it must not start or end with a hyphen, it cannot contain two consecutive hyphens, and it cannot be a 12 digit number.

For a list of Basic Latin (ASCII) characters, go to the [Library of Congress Basic Latin \(ASCII\) Code Table](#).

IAM Entity Object Limits

AWS allows you to request an increase to default IAM entity limits. To learn how to request a limit increase to these default limits, see [AWS Service Limits](#) in the *Amazon Web Services General Reference* documentation.

Default limits for IAM entities:

Resource	Default Limit
Customer managed policies in an AWS account	1500
Groups in an AWS account	300
Roles in an AWS account	1000
Users in an AWS account	5000 (If you need to add a large number of users, consider using temporary security credentials (p. 231) .)
Virtual MFA devices (assigned or unassigned) in an AWS account	Equal to the user quota for the account
Instance profiles in an AWS account	1000
Server certificates stored in an AWS account	20

For most IAM entity limits, you cannot request a limit increase.

Limits for IAM entities:

Resource	Limit
Access keys assigned to an IAM user	2
Access keys assigned to the AWS account root user	2
Aliases for an AWS account	1
Groups an IAM user can be a member of	10
IAM users in a group	Equal to the user quota for the account
Identity providers (IdPs) associated with an IAM SAML provider object	10
Keys per SAML provider	10
Login profiles for an IAM user	1

Resource	Limit
Managed policies attached to an IAM group	10
Managed policies attached to an IAM role	10
Managed policies attached to an IAM user	10
MFA devices in use by an IAM user	1
MFA devices in use by the AWS account root user	1
Roles in an instance profile	1
SAML providers in an AWS account	100
Signing certificates assigned to an IAM user	2
SSH public keys assigned to an IAM user	5
Versions of a managed policy that can be stored	5

IAM Entity Character Limits

The following are the maximum lengths for entities:

Description	Limit
Path	512 characters
User name	64 characters
Group name	128 characters
Role name	64 characters
Instance profile name	Important If you intend to use a role with the Switch Role feature in the AWS console, then the combined Path and RoleName cannot exceed 64 characters.
	128 characters
Unique IDs created by IAM, for example:	128 characters
<ul style="list-style-type: none"> • User IDs that begin with <code>AIDA</code> • Group IDs that begin with <code>AGPA</code> • Role IDs that begin with <code>AROA</code> • Managed policy IDs that begin with <code>ANPA</code> • Server certificate IDs that begin with <code>ASCA</code> 	
Note This is not intended to be an exhaustive list, nor is it a guarantee that IDs of a certain type begin only with the specified letter combination.	

Description	Limit
Policy name	128 characters
Password for a login profile	1 to 128 characters
Alias for an AWS account ID	3 to 63 characters
Role trust policy JSON text (the policy that determines who is allowed to assume the role)	2,048 characters
Role session name	64 characters
For inline policies (p. 279)	<p>You can add as many inline policies as you want to an IAM user, role, or group, but the total aggregate policy size (the sum size of all inline policies) per entity cannot exceed the following limits:</p> <ul style="list-style-type: none"> User policy size cannot exceed 2,048 characters Role policy size cannot exceed 10,240 characters Group policy size cannot exceed 5,120 characters <p>Note IAM does not count whitespace when calculating the size of a policy against these limitations.</p>
For managed policies (p. 279)	<ul style="list-style-type: none"> You can add up to 10 managed policies to an IAM user, role, or group. The size of each managed policy cannot exceed 6,144 characters. <p>Note IAM does not count whitespace when calculating the size of a policy against this limitation.</p>

AWS Services That Work with IAM

The AWS services listed below are grouped by their [AWS product categories](#) and include information about what IAM features they support:

- Service** – You can choose the name of a service to view the AWS documentation for that service.
- Actions** – You can specify individual actions in a policy. If the service does not support this feature, then **All actions** is selected in the [visual editor \(p. 319\)](#). In a JSON policy document, you must use * in the Action element. For a list of actions in each service, see [AWS Service Actions and Condition Context Keys for Use in IAM Policies \(p. 485\)](#).
- Resource-level permissions** – You can use ARNs to specify individual resources in the policy. If the service does not support this feature, then **All resources** is chosen in the [policy visual editor \(p. 319\)](#). In a JSON policy document, you must use * in the Resource element. Some actions, such as `List*`

actions, do not support specifying an ARN because they are designed to return multiple resources. If a service supports this feature for some resources but not others, it is indicated by yellow cells in the table. See the documentation for that service for more information.

- **Resource-based policies** – You can attach resource-based policies to a resource within the service. Resource-based policies include a Principal element to specify which IAM identities can access that resource. For more information, see [Identity-Based Policies and Resource-Based Policies \(p. 285\)](#).
- **Authorization based on tags** – You can use [resource tags](#) in the condition of a policy. For example, you might create a [policy that allows tag owners full access to Amazon RDS resources \(p. 311\)](#) that they have tagged. You do this by using a condition key such as rds:db-tag/Owner.
- **Temporary credentials** – Users signed in with federation, a cross-account role, or a [service role \(p. 135\)](#) can access the service. Temporary security credentials are obtained by calling AWS STS API operations like [AssumeRole](#) or [GetFederationToken](#). For more information, see [Temporary Security Credentials \(p. 231\)](#).
- **Service-linked roles** – A [service-linked role \(p. 136\)](#) gives the service permission to access resources in other services to complete an action on your behalf. Choose the Yes link to see the documentation for services that support these roles. For more information, see [Using Service-Linked Roles \(p. 176\)](#).
- **More information** – If a service doesn't fully support a feature, you can review the footnotes for an entry to view the limitations and links to related information.

Compute Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
Application Auto Scaling	Yes	Yes	No	No	Yes	Yes
Amazon EC2 Auto Scaling	Yes	Yes	No	No	Yes	Yes
AWS Batch	Yes	No	No	No	Yes	No
Amazon Elastic Compute Cloud (Amazon EC2)	Yes	Yes	No	Yes	Yes	Yes ¹
AWS Elastic Beanstalk	Yes	Yes	No	No	Yes	Yes
Amazon Elastic Container Registry (Amazon ECR)	Yes	Yes	Yes	No	Yes	No
Amazon Elastic Container Service (Amazon ECS)	Yes	Yes	No	No	Yes	Yes
Elastic Load Balancing	Yes	Yes	No	No	Yes	Yes
AWS Lambda	Yes	Yes	Yes ²	No	Yes	No
Amazon Lightsail	Yes	No	No	No	Yes	No

¹ Amazon EC2 service-linked roles cannot be created using the AWS Management Console, and can be used only for the following features: [Scheduled Instances](#), [Spot Instance Requests](#), [Spot Fleet Requests](#)

² The only AWS Lambda API action that can be specified in a resource-based policy is [lambda:InvokeFunction](#). For more information, see [Using Resource-Based Policies for AWS Lambda \(Lambda Function Policies\)](#) in the [AWS Lambda Developer Guide](#).

Storage and Migration Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
Amazon Elastic Block Store (Amazon EBS)	Yes	Yes	No	Yes	Yes	No
Amazon Elastic File System (Amazon EFS)	Yes	Yes	No	No	Yes	No
Amazon Glacier	Yes	Yes	Yes	Yes	Yes	No
AWS Import/Export	Yes	No	No	No	Yes	No
AWS Migration Hub	Yes	Yes	No	No	Yes	No
Amazon Simple Storage Service (Amazon S3)	Yes	Yes	Yes	Yes	Yes	No
AWS Snowball	Yes	No	No	No	Yes	No
AWS Snowball Edge	Yes	No	No	No	No	No
AWS Storage Gateway	Yes	Yes	No	No	Yes	No

Database Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
AWS Database Migration Service	Yes	Yes	No	Yes	Yes	No
Amazon DynamoDB	Yes	Yes	No	No	Yes	Yes
Amazon ElastiCache	Yes	No ¹	No	No	Yes	Yes
Amazon Redshift	Yes	Yes	No	Yes	Yes	Yes
Amazon Relational Database Service (Amazon RDS)	Yes	Yes	No	Yes	Yes	Yes
Amazon SimpleDB	Yes	Yes	No	No	Yes	No

¹ Two APIs specify an Amazon S3 ARN resource when seeding a cluster/replication group.

Networking and Content Delivery Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles

Amazon API Gateway	Yes	Yes	No	No	Yes	No
Amazon CloudFront	Yes ¹	No	No	No	Yes	No
AWS Direct Connect	Yes	No	No	No	Yes	No
Amazon Route 53	Yes	Yes	No	No	Yes	No
Amazon Virtual Private Cloud (Amazon VPC)	Yes	Yes ²	Yes ³	Yes	Yes	No

¹ CloudFront does not support action-level permissions for creating CloudFront key pairs. You must use an AWS account root user to create a CloudFront key pair. For more information, see [Creating CloudFront Key Pairs for Your Trusted Signers](#) in the *Amazon CloudFront Developer Guide*.

² In an IAM user policy, you cannot restrict permissions to a specific Amazon VPC endpoint. Any Action element that includes the ec2:*VpcEndpoint* or ec2:DescribePrefixLists API actions must specify ""Resource": "*". For more information, see [Controlling the Use of Endpoints](#) in the *Amazon VPC User Guide*.

³ Amazon VPC supports attaching a single resource policy to a VPC endpoint to restrict what can be accessed through that endpoint. For more information about using resource-based policies to control access to resources from specific Amazon VPC endpoints, see [Using Endpoint Policies](#) in the *Amazon VPC User Guide*.

Developer Tools and Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
AWS Cloud9	Yes	Yes	Yes	Yes	Yes	Yes
AWS CodeBuild	Yes	Yes	No	No	Yes	No
AWS CodeCommit	Yes	Yes	No	No	Yes	No
AWS CodeDeploy	Yes	Yes	No	No	Yes	No
AWS CodePipeline	Yes	Yes	No	No	Yes	No
AWS CodeStar	Yes	Yes ¹	No	No	No	No

Management Tools and Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
AWS CloudFormation	Yes	Yes	No	No	Yes	No
AWS CloudTrail	Yes	Yes	No	No	Yes	No
Amazon CloudWatch	Yes	No	No	No	Yes	Yes ¹

Amazon CloudWatch Events	Yes	Yes	No	No	Yes	No
Amazon CloudWatch Logs	Yes	Yes	No	No	Yes	No
AWS Config	Yes	No	No	No	Yes	No
AWS Health	Yes	No	No	No	Yes	No
AWS OpsWorks	Yes	Yes	Yes	No	Yes	No
AWS OpsWorks for Chef Automate	Yes	Yes	Yes	No	Yes	No
AWS Service Catalog	Yes	No	No	No	Yes	No
AWS Trusted Advisor	Yes ²	Yes	No	No	Yes ²	No

¹ Amazon CloudWatch service-linked roles cannot be created using the AWS Management Console, and support only the [Alarm Actions](#) feature.

² API access to Trusted Advisor is through the AWS Support API and is controlled by AWS Support IAM policies.

Media Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
Amazon Elastic Transcoder	Yes	Yes	No	No	Yes	No
AWS Elemental MediaPackage	Yes	Yes	No	No	Yes	No
AWS Elemental MediaStore	Yes	Yes	No	No	Yes	No
AWS Elemental MediaTailor	Yes	Yes	Yes	No	Yes	No
Kinesis Video Streams	Yes	Yes	No	No	No	No

Machine Learning Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
Amazon Comprehend	Yes	No	No	No	Yes	No
Amazon Lex	Yes	Yes	No	No	Yes	Yes
Amazon Machine Learning	Yes	Yes	No	Yes	Yes	No
Amazon Polly	Yes	Yes	No	No	Yes	No
Amazon Rekognition	Yes	Yes	No	No	No	No
Amazon SageMaker	Yes	Yes ¹	No	No	Yes	No

Amazon Transcribe	Yes	No	No	No	Yes	No
Amazon Translate	Yes	No	No	No	Yes	No

¹ Amazon SageMaker does not check external resources referenced in a call. For example, Amazon SageMaker does not support policies that restrict training jobs to certain AWS KMS keys or Amazon S3 buckets.

Analytics Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
Amazon CloudSearch	Yes	Yes	No	No	Yes	No
AWS Data Pipeline	Yes	No	No	Yes	Yes	No
Amazon Elasticsearch Service	Yes	Yes	Yes	No	Yes	Yes
Amazon EMR	Yes	No	No	Yes	Yes	Yes
AWS Glue	Yes	No	No	No	No	No
Amazon Kinesis Data Analytics	Yes	Yes	No	No	Yes	No
Amazon Kinesis Data Firehose	Yes	Yes	No	No	Yes	No
Amazon Kinesis Data Streams	Yes	Yes	No	No	Yes	No
Amazon QuickSight	Yes	No	No	No	No	No

Security, Identity, and Compliance Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
AWS Artifact	Yes	Yes	No	No	Yes	No
AWS Certificate Manager (ACM)	Yes	Yes	No	No	Yes	No
AWS CloudHSM	Yes	No	No	No	Yes	No
AWS CloudHSM Classic	Yes	No	No	No	No	No
Amazon Cognito	Yes	Yes	No	No	Yes	No
AWS Directory Service	Yes	No	No	No	Yes	No
Amazon GuardDuty	Yes	Yes	No	No	No	Yes ¹
AWS Identity and Access Management (IAM)	Yes	Yes	No	No	Yes ²	No
Amazon Inspector	Yes	No	No	No	Yes	Yes

AWS Key Management Service (AWS KMS)	Yes	Yes	Yes	No	Yes	No
AWS Organizations	Yes	Yes	No	No	Yes	Yes
AWS Single Sign-On (AWS SSO)	Yes	No	Yes	No	Yes	No
AWS Security Token Service (AWS STS)	Yes	Yes ³	No	No	Yes ⁴	No
AWS Shield Advanced	Yes	No	No	No	Yes	No
AWS WAF	Yes	Yes	No	No	Yes	No

¹ GuardDuty does not support creating a service-linked role using the IAM console.

² Only some of the API actions for IAM can be called with temporary credentials. For more information, see [Comparing your API options](#)

³ AWS STS does not have "resources," but does allow restricting access in a similar way to users. For more information, see [Denying Access to Temporary Security Credentials by Name](#).

⁴ Only some of the APIs for AWS STS support calling with temporary credentials. For more information, see [Comparing your API options](#).

Mobile Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
AWS Device Farm	Yes	No	No	No	Yes	No
Amazon Mobile Analytics	Yes	No	No	No	Yes	No
AWS Mobile Hub	Yes	Yes	No	No	Yes	No
Amazon Pinpoint	Yes	Yes	No	No	Yes	No

Application Integration Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
Amazon MQ	Yes	No	No	No	Yes	No
Amazon Simple Email Service (Amazon SES)	Yes	Yes ¹	No	No	Yes ²	No
Amazon Simple Notification Service (Amazon SNS)	Yes	Yes	Yes	No	Yes	No
Amazon Simple Queue Service (Amazon SQS)	Yes	Yes	Yes	No	Yes	No

Amazon Simple Workflow Service (Amazon SWF)	Yes	Yes	No	Yes	Yes	No
---	-----	-----	----	-----	-----	----

¹ Amazon SES supports resource-level permissions in policies that grant permissions to delegate senders to access specific SES identities.

² Only the Amazon SES API supports temporary security credentials. The Amazon SES SMTP interface does not support SMTP credentials that are derived from temporary security credentials.

Business Productivity Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
Alexa for Business	Yes	Yes	No	Yes	Yes	No
Amazon WorkDocs	Yes	No	No	No	Yes	No
Amazon WorkMail	Yes	No	No	No	Yes	No

Desktop and App Streaming Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
Amazon AppStream	Yes	No	No	No	Yes	No
Amazon AppStream 2.0	Yes	No	No	No	Yes	No
Amazon WorkSpaces	Yes	Yes	No	No	Yes	No
Amazon WAM	Yes	No	No	No	Yes	No

Internet of Things Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
AWS Greengrass	Yes	Yes	Yes	No	Yes	No
AWS IoT	Yes	Yes	Yes ¹	No	Yes	No

¹ Devices connected to AWS IoT are authenticated by using X.509 certificates. You can attach AWS IoT policies to an X.509 certificate to control what the device is authorized to do. For more information, see [Create an AWS IoT Policy](#) in the *AWS IoT Developer Guide*.

Game Development Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
Amazon GameLift	Yes	No	No	No	Yes	No

Software Services

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
AWS Marketplace	Yes	Yes	No	No	Yes	No

Additional Resources

Service	Actions	Resource-level permissions	Resource-based policies	Authorization based on tags	Temporary credentials	Service-linked roles
AWS Billing and Cost Management	Yes	No	No	No	Yes	No
AWS Support	No	No	No	No	Yes	No

IAM JSON Policy Reference

This section presents detailed syntax, descriptions, and examples of the elements, variables, and evaluation logic of JSON policies in IAM. It includes the following sections.

- [IAM JSON Policy Elements Reference \(p. 426\)](#) — Learn more about the elements that you can use when you create a policy. View additional policy examples and learn about conditions, supported data types, and how they are used in various services.
- [IAM JSON Policy Evaluation Logic \(p. 458\)](#) — This section describes AWS requests, how they are authenticated, and how AWS uses policies to determine access to resources.
- [Grammar of the IAM JSON Policy Language \(p. 463\)](#) — This section presents a formal grammar for the language used to create policies in IAM.
- [AWS Managed Policies for Job Functions \(p. 468\)](#) — This section lists all of the AWS managed policies that directly map to common job functions in the IT industry. Use these policies to grant the permissions needed to carry out the tasks expected of someone in a specific job function. These policies consolidate permissions for many services into a single policy.
- [AWS Service Actions and Condition Context Keys for Use in IAM Policies \(p. 485\)](#) — This section presents a list of all of the AWS API actions that can be used as permissions in an IAM policy and the service-specific condition keys that can be used to further refine the request.
- [IAM Policy Actions Grouped by Access Level \(p. 615\)](#) — This section presents a list the access levels that all AWS API actions are members of. Each API action that can be used as a policy permission is categorized into one access level.

Important

You cannot save any policy that does not comply with the established policy syntax. You can use Policy Validator to detect and correct invalid policies. One click takes you to an editor that shows both the existing policy and a copy with the recommended changes. You can accept the changes or make further modifications. For more information, see [Validating JSON Policies \(p. 321\)](#).

IAM JSON Policy Elements Reference

JSON policy documents are made up of elements. The elements are listed here in the general order you use them in a policy. The order of the elements doesn't matter—for example, the Resource element can come before the Action element. You're not required to specify any Condition elements in the policy.

The details of what goes into a policy vary for each service, depending on what actions the service makes available, what types of resources it contains, and so on. When you're writing policies for a specific service, it's helpful to see examples of policies for that service. For a list of all the services that support IAM, and for links to the documentation in those services that discusses IAM and policies, see [AWS Services That Work with IAM \(p. 417\)](#).

Topics

- [IAM JSON Policy Elements: Version \(p. 426\)](#)
- [IAM JSON Policy Elements: Id \(p. 427\)](#)
- [IAM JSON Policy Elements: Statement \(p. 427\)](#)
- [IAM JSON Policy Elements: Sid \(p. 428\)](#)
- [IAM JSON Policy Elements: Effect \(p. 428\)](#)
- [IAM JSON Policy Elements: Principal \(p. 428\)](#)
- [IAM JSON Policy Elements: NotPrincipal \(p. 432\)](#)
- [IAM JSON Policy Elements: Action \(p. 434\)](#)
- [IAM JSON Policy Elements: NotAction \(p. 435\)](#)
- [IAM JSON Policy Elements: Resource \(p. 436\)](#)
- [IAM JSON Policy Elements: NotResource \(p. 437\)](#)
- [IAM JSON Policy Elements: Condition \(p. 438\)](#)
- [IAM Policy Elements: Variables \(p. 451\)](#)
- [IAM JSON Policy Elements: Supported Data Types \(p. 457\)](#)

IAM JSON Policy Elements: Version

Disambiguation Note

This topic is about the JSON policy element named "Version". The `version` policy element is different from a *policy version*. The `version` policy element is used within a policy and defines the version of the policy language. A policy version, on the other hand, is created when you make changes to a customer managed policy in IAM. The changed policy doesn't overwrite the existing policy. Instead, IAM creates a new version of the managed policy. If you were searching for information about the multiple version support available for managed policies, see the section called "[Versioning IAM Policies](#) (p. 335)".

The `Version` elements specifies the language syntax rules that are to be used to process this policy. If you include features that are not available in the specified version, then your policy will generate errors or not work the way you intend. As a general rule, you should specify the most recent version available, unless you depend on a feature that was deprecated in later versions.

The `Version` element must appear before the `Statement` element. The only allowed values are these:

- 2012-10-17. This is the current version of the policy language, and you should use this version number for all policies.
- 2008-10-17. This was an earlier version of the policy language. You might see this version on existing policies. Do not use this version for any new policies or any existing policies that you are updating.

If you do not include a `Version` element, the value defaults to 2008-10-17. However, it is a good practice to always include a `Version` element and set it to 2012-10-17.

Note

If your policy uses any features that were introduced in a later version, such as [policy variables \(p. 451\)](#), you *must* include a `Version` element and set it to 2012-10-17. If you don't include a `Version` element set to 2012-10-17, variables such as `#{aws:username}` aren't recognized as variables and are instead treated as literal strings in the policy.

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

IAM JSON Policy Elements: Id

The `Id` element specifies an optional identifier for the policy. The ID is used differently in different services.

For services that let you set an `Id` element, we recommend you use a UUID (GUID) for the value, or incorporate a UUID as part of the ID to ensure uniqueness.

```
"Id": "cd3ad3d9-2776-4ef1-a904-4c229d1642ee"
```

Note

Some AWS services (for example, Amazon SQS or Amazon SNS) might require this element and have uniqueness requirements for it. For service-specific information about writing policies, refer to the documentation for the service you're working with.

IAM JSON Policy Elements: Statement

The `Statement` element is the main element for a policy. This element is required. It can include multiple elements (see the subsequent sections in this page). The `Statement` element contains an array of individual statements. Each individual statement is a JSON block enclosed in braces { }.

```
"Statement": [{...},{...},{...}]
```

The following example shows a policy that contains an array of three statements inside a single `Statement` element. (The policy allows you to access your own "home folder" in the Amazon S3 console.) The policy includes the `aws:username` variable, which is replaced during policy evaluation with the user name from the request. For more information, see [Introduction \(p. 451\)](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3>ListAllMyBuckets",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::*"
    }
  ]
}
```

```
{  
    "Effect": "Allow",  
    "Action": "s3>ListBucket",  
    "Resource": "arn:aws:s3:::BUCKET-NAME",  
    "Condition": {"StringLike": {"s3:prefix": [  
        "",  
        "home/",  
        "home/${aws:username}/"  
    ]}}  
},  
{  
    "Effect": "Allow",  
    "Action": "s3:*",  
    "Resource": [  
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}",  
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}/*"  
    ]  
}  
]
```

IAM JSON Policy Elements: Sid

The `Sid` (statement ID) is an optional identifier that you provide for the policy statement. You can assign a `Sid` value to each statement in a statement array. In services that let you specify an `ID` element, such as SQS and SNS, the `Sid` value is just a sub-ID of the policy document's ID. In IAM, the `Sid` value must be unique within a JSON policy.

```
"Sid": "1"
```

In IAM, the `Sid` is not exposed in the IAM API. You can't retrieve a particular statement based on this ID.

Note

Some AWS services (for example, Amazon SQS or Amazon SNS) might require this element and have uniqueness requirements for it. For service-specific information about writing policies, refer to the documentation for the service you're working with.

IAM JSON Policy Elements: Effect

The `Effect` element is required and specifies whether the statement results in an allow or an explicit deny. Valid values for `Effect` are `Allow` and `Deny`.

```
"Effect": "Allow"
```

By default, access to resources is denied. To allow access to a resource, you must set the `Effect` element to `Allow`. To override an allow (for example, to override an allow that is otherwise in force), you set the `Effect` element to `Deny`. For more information, see [IAM JSON Policy Evaluation Logic \(p. 458\)](#).

IAM JSON Policy Elements: Principal

Use the `Principal` element to specify the user (IAM user, federated user, or assumed-role user), AWS account, AWS service, or other principal entity that is allowed or denied access to a resource. You use the `Principal` element in the trust policies for IAM roles and in resource-based policies—that is, in policies that you embed directly in a resource. For example, you can embed such policies in an Amazon S3 bucket, an Amazon Glacier vault, an Amazon SNS topic, an Amazon SQS queue, or an AWS KMS customer master key (CMK).

Use the `Principal` element in these ways:

- In IAM roles, use the `Principal` element in the role's trust policy to specify who can assume the role. For cross-account access, you must specify the 12-digit identifier of the trusted account.

Note

After you create the role, you can change the account to "*" to allow everyone to assume the role. If you do this, we strongly recommend that you limit who can access the role through other means, such as a `Condition` element that limits access to only certain IP addresses. Do not leave your role accessible to everyone!

- In resource-based policies, use the `Principal` element to specify the accounts or users who are allowed to access the resource.

Do not use the `Principal` element in policies that you attach to IAM users and groups. Similarly, you do not specify a principal in the permission policy for an IAM role. In those cases, the principal is implicitly the user that the policy is attached to (for IAM users) or the user who assumes the role (for role access policies). When the policy is attached to an IAM group, the principal is the IAM user in that group who is making the request.

Specifying a Principal

You specify a principal using the [Amazon Resource Name \(ARN\)](#) (p. 410) of the AWS account, IAM user, IAM role, federated user, or assumed-role user. You cannot specify IAM groups as principals. When you specify an AWS account, you can use a shortened form that consists of the `AWS:` prefix followed by the account ID, instead of using the account's full ARN. In the AWS Management Console, specify only the 12-digit account ID.

The following examples show various ways in which principals can be specified.

Specific AWS accounts

When you use an AWS account identifier as the principal in a policy, the permissions in the policy statement can be granted to all identities contained in that account. This includes IAM users and roles in that account. The following examples show different ways to specify an AWS account as a principal.

```
"Principal": { "AWS": "arn:aws:iam::AWS-account-ID:root" }
```

```
"Principal": { "AWS": "AWS-account-ID" }
```

You can specify more than one AWS account as a principal, as shown in the following example.

```
"Principal": {  
    "AWS": [  
        "arn:aws:iam::AWS-account-ID:root",  
        "arn:aws:iam::AWS-account-ID:root"  
    ]  
}
```

Individual IAM user or users

You can specify an individual IAM user (or array of users) as the principal, as in the following examples.

Note

In a `Principal` element, the user name is case sensitive.

```
"Principal": { "AWS": "arn:aws:iam::AWS-account-ID:user/user-name" }
```

```
"Principal": {  
    "AWS": [
```

```
    "arn:aws:iam::AWS-account-ID:user/user-name-1",  
    "arn:aws:iam::AWS-account-ID:user/UserName2"  
]  
}
```

When you specify users in a `Principal` element, you cannot use a wildcard (*) to mean "all users". Principals must always name a specific user or users.

Important

If your `Principal` element in a role trust policy contains an ARN that points to a specific IAM user, then that ARN is transformed to the user's unique principal ID when the policy is saved. This helps mitigate the risk of someone escalating their privileges by removing and recreating the user. You don't normally see this ID in the console, because there is also a reverse transformation back to the user's ARN when the trust policy is displayed. However, if you delete the user, then the relationship is broken. The policy no longer applies, even if you recreate the user. That's because the new user has a new principal ID that does not match the ID stored in the trust policy. When this happens, the principal ID shows up in the console because AWS can no longer map it back to a valid ARN. The result is that if you delete and recreate a user referenced in a trust policy's `Principal` element, you must edit the role to replace the now incorrect principal ID with the correct ARN. The ARN is once again transformed into the user's new principal ID when you save the policy.

Federated users (using web identity federation)

```
"Principal": { "Federated": "cognito-identity.amazonaws.com" }
```

```
"Principal": { "Federated": "www.amazon.com" }
```

```
"Principal": { "Federated": "graph.facebook.com" }
```

```
"Principal": { "Federated": "accounts.google.com" }
```

Federated users (using a SAML identity provider)

```
"Principal": { "Federated": "arn:aws:iam::AWS-account-ID:saml-provider/provider-name" }
```

IAM role

```
"Principal": { "AWS": "arn:aws:iam::AWS-account-ID:role/role-name" }
```

Important

If your `Principal` element in a role trust policy contains an ARN that points to a specific IAM role, then that ARN is transformed to the role's unique principal ID when the policy is saved. This helps mitigate the risk of someone escalating their privileges by removing and recreating the role. You don't normally see this ID in the console, because there is also a reverse transformation back to the role's ARN when the trust policy is displayed. However, if you delete the role, then the relationship is broken. The policy no longer applies, even if you recreate the role because the new role has a new principal ID that does not match the ID stored in the trust policy. When this happens, the principal ID shows up in the console because AWS can no longer map it back to a valid ARN. The end result is that if you delete and recreate a role referenced in a trust policy's `Principal` element, you must edit the role to replace the now incorrect principal ID with the correct ARN. The ARN will once again be transformed into the role's new principal ID when you save the policy.

Specific assumed-role user

```
"Principal": { "AWS": "arn:aws:sts::AWS-account-ID:assumed-role/role-name/role-session-name" }
```

AWS service

IAM roles that can be assumed by an AWS service are called [service roles \(p. 135\)](#). Service roles must include a trust policy. *Trust policies* are resource-based policies that are attached to a role that define which principals can assume the role. Some service role have predefined trust policies. However, in some cases, you must specify the service principal in the trust policy. A *service principal* is an identifier that is used to grant permissions to a service. The identifier includes the long version of a service name, and is usually in the following format:

long_service-name.amazonaws.com

The service principal is defined by the service. To learn the service principal for a service, see the documentation for that service.

The following example shows a policy that can be attached to a service role. The policy enables two services, Amazon EMR and AWS Data Pipeline, to assume the role. The services can then perform any tasks granted by the permissions policy assigned to the role (not shown). To specify multiple service principals, you do not specify two *Service* elements; you can have only one. Instead, you use an array of multiple service principals as the value of a single *Service* element.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "elasticmapreduce.amazonaws.com",
          "datapipeline.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Some AWS services support additional options for specifying a principal. For example, Amazon S3 lets you specify a [canonical user ID](#) using the following format:

```
"Principal": { "CanonicalUser": "79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be" }
```

Everyone (anonymous users)

The following are equivalent:

```
"Principal": "*"
```

```
"Principal" : { "AWS" : "*" }
```

Note

In these examples, the asterisk (*) is used as a placeholder for Everyone/Anonymous. You cannot use it as a wildcard to match part of a name or an ARN. We also strongly recommend that you do not use a wildcard in the *Principal* element in a role's trust policy unless you otherwise

restrict access through a `Condition` element in the policy. Otherwise, any IAM user in any account can access the role.

More Information

For more information, see the following:

- [Bucket Policy Examples](#) in the *Amazon Simple Storage Service Developer Guide*
- [Example Policies for Amazon SNS](#) in the *Amazon Simple Notification Service Developer Guide*
- [Amazon SQS Policy Examples](#) in the *Amazon Simple Queue Service Developer Guide*
- [Key Policies](#) in the *AWS Key Management Service Developer Guide*
- [Account Identifiers](#) in the *AWS General Reference*
- [About Web Identity Federation](#) (p. 143)

IAM JSON Policy Elements: NotPrincipal

Use the `NotPrincipal` element to specify an exception to a list of principals. For example, you can deny access to all principals *except* the one named in the `NotPrincipal` element. The syntax for specifying `NotPrincipal` is the same as for specifying [IAM JSON Policy Elements: Principal](#) (p. 428).

Note that you can also use `NotPrincipal` to allow all principals *except* the one named in the `NotPrincipal` element; however, we do not recommend this.

Warning

When you use `NotPrincipal` in the same policy statement as "`Effect`": "Allow", the permissions specified in the policy statement will be granted to *all* principals except for the one(s) specified, including anonymous (unauthenticated) users. We strongly recommend you do not use `NotPrincipal` in the same policy statement as "`Effect`": "Allow".

When you use `NotPrincipal` in the same policy statement as "`Effect`": "Deny", the permissions specified in the policy statement are explicitly denied to all principals *except* for the one(s) specified. This enables you to implement a form of "whitelisting". When you explicitly deny access to an AWS account, you deny access to all users contained in that account.

Note that if a resource-based policy combines "`Effect`": "Deny" with a `NotPrincipal` element that specifies only a principal in an AWS account, the policy is likely denying access to the account containing the principal, and that in turn results in the specified principal not being able to access the resource. To understand how this can happen, see the examples in the next section.

Important

Very few scenarios require the use of `NotPrincipal`, and we recommend that you explore other authorization options before you decide to use `NotPrincipal`.

Specifying `NotPrincipal` in the same policy statement as "`Effect`": "Deny"

You specify principals in the `NotPrincipal` element using the same syntax that you use for specifying principals in the [IAM JSON Policy Elements: Principal](#) (p. 428) element. However, it can be difficult to achieve the intended effect, particularly when you combine `NotPrincipal` with "`Effect`": "Deny" in the same policy statement and you work across AWS account boundaries.

Important

Combining Deny and `NotPrincipal` is the only time that the order in which AWS evaluates principals makes a difference. AWS internally validates the principals from the "top down," meaning that AWS checks the account first and then the user. If an assumed-role user (someone who is using a role rather than an IAM user) is being evaluated, AWS looks at the account first, then the role, and finally the assumed-role user. The assumed-role user is identified by the role session name that is specified when the user assumes the role.

Normally, this order does not have any impact on the results of the policy evaluation. However, when you use both Deny and NotPrincipal, the evaluation order requires you to explicitly include the ARNs for the entities associated with the specified principal. For example, to specify a user, you must explicitly include the ARN for the user's account. To specify an assumed-role user, you must also include both the ARN for the role and the ARN for the account containing the role.

The following examples show how to use NotPrincipal and "Effect": "Deny" in the same policy statement effectively.

Example 1: An IAM user in the same or a different account

In the following example, all principals *except* the user named Bob in AWS account 444455556666 are explicitly denied access to a resource. Note that to achieve the intended effect, the NotPrincipal element contains the ARN of both the user Bob and the AWS account that Bob belongs to (`arn:aws:iam::444455556666:root`). If the NotPrincipal element contained only Bob's ARN, the effect of the policy would be to explicitly deny access to the AWS account that contains the user Bob. A user cannot have more permissions than its parent account, so if Bob's account is explicitly denied access then Bob is also unable to access the resource.

This example works as intended when it is part of a policy statement in a resource-based policy that is attached to a resource in either the same or a different AWS account (not 444455556666). This example by itself does not grant access to Bob, it only omits Bob from the list of principals that are explicitly denied. To give Bob access to the resource, another policy statement must explicitly allow access using "Effect": "Allow".

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "NotPrincipal": {"AWS": [
                "arn:aws:iam::444455556666:user/Bob",
                "arn:aws:iam::444455556666:root"
            ]},
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3::::BUCKETNAME",
                "arn:aws:s3::::BUCKETNAME/*"
            ]
        }
    ]
}
```

Example 2: An IAM role in the same or different account

In the following example, all principals *except* the assumed-role user named cross-account-audit-app in AWS account 444455556666 are explicitly denied access to a resource. Note that to achieve the intended effect, the NotPrincipal element contains the ARN of the assumed-role user (cross-account-audit-app), the role (cross-account-read-only-role), and the AWS account that the role belongs to (444455556666). If the NotPrincipal element was missing the ARN of the role, the effect of the policy would be to explicitly deny access to the role. Similarly, if the NotPrincipal element was missing the ARN of the AWS account that the role belongs to, the effect of the policy would be to explicitly deny access to the AWS account and all entities in that account. Assumed-role users cannot have more permissions than their parent role, and roles cannot have more permissions than their parent AWS account, so when the role or the account is explicitly denied access, the assumed role user is unable to access the resource.

This example works as intended when it is part of a policy statement in a resource-based policy that is attached to a resource in a different AWS account (not 444455556666). This example by itself does not grant access to the assumed-role user cross-account-audit-app, it only omits cross-account-audit-

app from the list of principals that are explicitly denied. To give cross-account-audit-app access to the resource, another policy statement must explicitly allow access using "Effect": "Allow".

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "NotPrincipal": {"AWS": [
                "arn:aws:sts::444455556666:assumed-role/cross-account-read-only-role/cross-
account-audit-app",
                "arn:aws:iam::444455556666:role/cross-account-read-only-role",
                "arn:aws:iam::444455556666:root"
            ]},
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::Bucket_AccountAudit",
                "arn:aws:s3:::Bucket_AccountAudit/*"
            ]
        }
    ]
}
```

IAM JSON Policy Elements: Action

The Action element describes the specific action or actions that will be allowed or denied. Statements must include either an Action or NotAction element. Each AWS service has its own set of actions that describe tasks that you can perform with that service. For example, the list of actions for Amazon S3 can be found at [Specifying Permissions in a Policy](#) in the *Amazon Simple Storage Service Developer Guide*, the list of actions for Amazon EC2 can be found in the [Amazon EC2 API Reference](#), and the list of actions for AWS Identity and Access Management can be found in the [IAM API Reference](#). To find the list of actions for other services, consult the API reference [documentation](#) for the service.

You specify a value using a namespace that identifies a service (iam, ec2, sqs, sns, s3, etc.) followed by the name of the action to allow or deny. The name must match an action that is supported by the service. The prefix and the action name are case insensitive. For example, iam>ListAccessKeys is the same as IAM:listaccesskeys. The following examples show Action elements for different services.

Amazon SQS action

```
"Action": "sqs:SendMessage"
```

Amazon EC2 action

```
"Action": "ec2:StartInstances"
```

IAM action

```
"Action": "iam:ChangePassword"
```

Amazon S3 action

```
"Action": "s3:GetObject"
```

You can specify multiple values for the Action element.

```
"Action": [ "sqs:SendMessage", "sqs:ReceiveMessage", "ec2:StartInstances",
"iam:ChangePassword", "s3:GetObject" ]
```

You can use a wildcard (*) to give access to all the actions the specific AWS product offers. For example, the following Action element applies to all S3 actions.

```
"Action": "s3:/*"
```

You can also use wildcards (*) as part of the action name. For example, the following Action element applies to all IAM actions that include the string AccessKey, including CreateAccessKey, DeleteAccessKey, ListAccessKeys, and UpdateAccessKey.

```
"Action": "iam:*AccessKey*"
```

Some services let you limit the actions that are available. For example, Amazon SQS lets you make available just a subset of all the possible Amazon SQS actions. In that case, the * wildcard doesn't allow complete control of the queue; it allows only the subset of actions that you've shared. For more information, see [Understanding Permissions](#) in the *Amazon Simple Queue Service Developer Guide*.

IAM JSON Policy Elements: NotAction

NotAction is an advanced policy element that explicitly matches everything *except* the specified list of actions. Using NotAction can result in a shorter policy by listing only a few actions that should not match, rather than including a long list of actions that will match. When using NotAction, you should keep in mind that actions specified in this element are the *only* actions in that are limited. This, in turn, means that all of the applicable actions or services that are not listed are allowed if you use the Allow effect, or are denied if you use the Deny effect. When you use NotAction with the Resource element, you provide scope for the policy. This is how AWS determines which actions or services are applicable. For more information, see the following example policy.

NotAction with Allow

You can use the NotAction element in a statement with "Effect": "Allow" to provide access to all of the actions in an AWS service, except for the actions specified in NotAction. You can use it with the Resource element to provide scope for the policy, limiting the allowed actions to the actions that can be performed on the specified resource.

The following example allows users to access all of the Amazon S3 actions that can be performed on any S3 resource *except* for deleting a bucket. This does not allow users to use the ListAllMyBuckets S3 API operation, because that action requires the "*" resource. This policy also does not allow actions in other services, because other service actions are not applicable to the S3 resources.

```
"Effect": "Allow",
"NotAction": "s3>DeleteBucket",
"Resource": "arn:aws:s3:::/*",
```

Sometimes, you might want to allow access to a large number of actions, and by using the NotAction element you effectively reverse the statement, resulting in a shorter list of actions. For example, because there are a lot of AWS services, you might want to create a policy that allows the user to do everything *except* access IAM actions.

The following example allows users to access every action in every AWS service *except* for IAM.

```
"Effect": "Allow",
"NotAction": "iam:*",
"Resource": "*"
```

Be careful using the NotAction element and "Effect": "Allow" in the same statement or in a different statement within a policy. NotAction matches all services and actions that are not explicitly

listed or applicable to the specified resource, and could result in granting users more permissions than you intended.

NotAction with Deny

You can use the `NotAction` element in a statement with `"Effect": "Deny"` to deny access to all of the listed resources except for the actions specified in the `NotAction` element. This combination does not allow the listed items, but instead explicitly denies the actions not listed. You must still allow actions that you want to allow.

The following conditional example denies access to non-IAM actions if the user is not signed-in using MFA. If the user is signed-in with MFA, then the `"Condition"` test fails and the final `"Deny"` statement has no effect. Note, however, that this would not grant the user access to any actions, it would only explicitly deny all other actions except IAM actions.

```
"Effect": "Deny",
"NotAction": "iam:*",
"Resource": "*",
"Condition": { "BoolIfExists": { "aws:MultiFactorAuthPresent": "false" }}
```

IAM JSON Policy Elements: Resource

The `Resource` element specifies the object or objects that the statement covers. Statements must include either a `Resource` or a `NotResource` element. You specify a resource using an ARN. (For more information about the format of ARNs, see [Amazon Resource Names \(ARNs\) and AWS Service Namespaces](#).)

Each service has its own set of resources. Although you always use an ARN to specify a resource, the details of the ARN for a resource depend on the service and the resource. For information about how to specify a resource, refer to the documentation for the service whose resources you're writing a statement for.

Note

Some services do not let you specify actions for individual resources; instead, any actions that you list in the `Action` or `NotAction` element apply to all resources in that service. In these cases, you use the wildcard `*` in the `Resource` element.

The following example refers to a specific Amazon SQS queue.

```
"Resource": "arn:aws:sqs:us-east-2:account-ID-without-hyphens:queue1"
```

The following example refers to the IAM user named Bob in an AWS account.

```
"Resource": "arn:aws:iam::account-ID-without-hyphens:user/Bob"
```

You can use wildcards as part of the resource ARN. You can use wildcard characters (`*` and `?`) within any ARN segment (the parts separated by colons). An asterisk (`*`) represents any combination of characters and a question mark (`?`) represents any single character. You can have use multiple `*` or `?` characters in each segment, but a wildcard cannot span segments. The following example refers to all IAM users whose path is `/accounting`.

```
"Resource": "arn:aws:iam::account-ID-without-hyphens:user/accounting/*"
```

The following example refers to all items within a specific Amazon S3 bucket.

```
"Resource": "arn:aws:s3:::my_corporate_bucket/*"
```

You can specify multiple resources. The following example refers to two DynamoDB tables.

```
"Resource": [
    "arn:aws:dynamodb:us-east-2:account-ID-without-hyphens:table/books_table",
    "arn:aws:dynamodb:us-east-2:account-ID-without-hyphens:table/magazines_table"
]
```

In the `Resource` element, you can use JSON [policy variables \(p. 451\)](#) in the part of the ARN that identifies the specific resource (that is, in the trailing part of the ARN). For example, you can use the key `{aws:username}` as part of a resource ARN to indicate that the current user's name should be included as part of the resource's name. The following example shows how you can use the `{aws:username}` key in a `Resource` element. The policy allows access to a Amazon DynamoDB table that matches the current user's name.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "dynamodb:*",
        "Resource": "arn:aws:dynamodb:us-east-2:ACCOUNT-ID-WITHOUT-HYPHENS:table/
${aws:username}"
    }
}
```

For more information about JSON policy variables, see [IAM Policy Elements: Variables \(p. 451\)](#).

IAM JSON Policy Elements: NotResource

`NotResource` is an advanced policy element that explicitly matches everything except the specified list of resources. Using `NotResource` can result in a shorter policy by listing only a few resources that should not match, rather than including a long list of resources that will match. When using `NotResource`, you should keep in mind that resources specified in this element are the *only* resources that are limited. This, in turn, means that all of the resources, including the resources in all other services, that are not listed are allowed if you use the `Allow` effect, or are denied if you use the `Deny` effect. Statements must include either a `Resource` or a `NotResource` element that specifies a resource using an ARN. (For more information about the format of ARNs, see [Amazon Resource Names \(ARNs\) and AWS Service Namespaces](#).)

Be careful using the `NotResource` element and `"Effect": "Allow"` in the same statement or in a different statement within a policy. `NotResource` allows all services and resources that are not explicitly listed, and could result in granting users more permissions than you intended. Using the `NotResource` element and `"Effect": "Deny"` in the same statement denies services and resources that are not explicitly listed.

For example, imagine you have a group named `HRPayroll`. Members of `HRPayroll` should not be allowed to access any Amazon S3 resources except the `Payroll` folder in the `HRBucket` bucket. The following policy explicitly denies access to all Amazon S3 resources other than the listed resources. Note, however, that this policy does not grant the user access to any resources.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "s3:*",
        "NotResource": [
            "arn:aws:s3:::HRBucket/Payroll",
            "arn:aws:s3:::HRBucket/Payroll/*"
        ]
}
```

```
}
```

Normally, to explicitly deny access to a resource you would write a policy that uses "Effect": "Deny" and that includes a Resource element that lists each folder individually. However, in that case, each time you add a folder to HRBucket, or add a resource to Amazon S3 that should not be accessed, you must add its name to the list in Resource. If you use a NotResource element instead, users are automatically denied access to new folders unless you add the folder names to the NotResource element.

IAM JSON Policy Elements: Condition

The Condition element (or Condition *block*) lets you specify conditions for when a policy is in effect. The Condition element is optional. In the Condition element, you build expressions in which you use [condition operators \(p. 440\)](#) (equal, less than, etc.) to match the condition in the policy against values in the request. Condition values can include date, time, the IP address of the requester, the ARN of the request source, the user name, user ID, and the user agent of the requester. Some services let you specify additional values in conditions; for example, Amazon S3 lets you write a condition using the s3:VersionId key, which is unique to that service.

- For a list of all of the condition operators and a description of how each one works, see [Condition Operators \(p. 440\)](#)
- For a description of how to handle condition keys that have multiple values, see [Creating a Condition That Tests Multiple Key Values \(Set Operations\) \(p. 447\)](#)
- For a list of all of the globally available condition keys, see [Available Global Condition Keys \(p. 477\)](#).
- For conditions keys that are defined by each service, see [AWS Service Actions and Condition Context Keys for Use in IAM Policies \(p. 485\)](#).

The Condition Block

The following example shows the basic format of a Condition element:

```
"Condition": {  
    "DateGreaterThan" : {  
        "aws:CurrentTime" : "2013-12-15T12:00:00Z"  
    }  
}
```

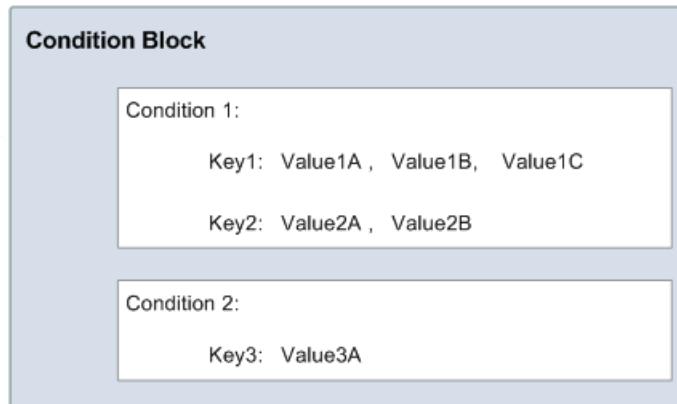
A value from the request is represented by a key, in this case aws:CurrentTime. The key value is compared to a value that you specify either as a literal value (2013-08-16T12:00:00Z) or as a policy variable, as explained later. The type of comparison to make is specified by the [condition operator \(p. 440\)](#) (here, DateGreaterThan). You can create conditions that compare strings, dates, numbers, and so on, using typical boolean comparisons like equals, greater than, and less than.

Under some circumstances, keys can contain multiple values. For example, a request to DynamoDB might ask to return or update multiple attributes from a table. A policy for access to DynamoDB tables can include the dynamodb:Attributes key, which contains all the attributes listed in the request. You can test the multiple attributes in the request against a list of allowed attributes in a policy by using set operators in the Condition element. For more information, see [Creating a Condition That Tests Multiple Key Values \(Set Operations\) \(p. 447\)](#).

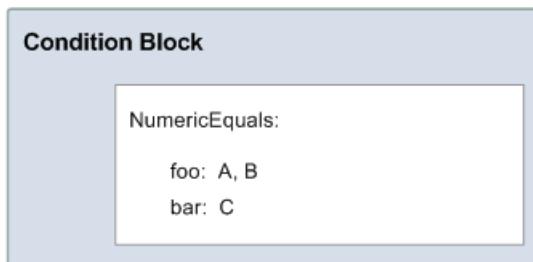
When the policy is evaluated during a request, AWS replaces the key with the corresponding value from the request. (In this example, AWS would use the date and time of the request.) The condition is evaluated to return true or false, which is then factored into whether the policy as a whole allows or denies the request.

Multiple Values in a Condition

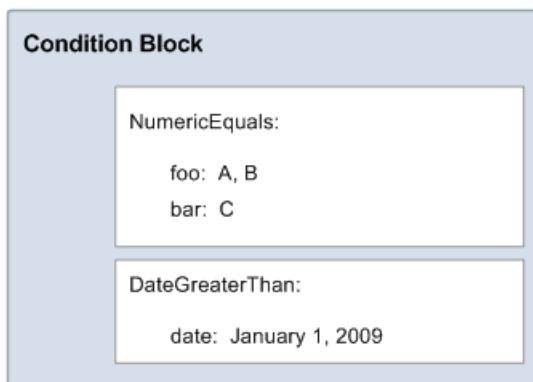
A Condition element can contain multiple conditions, and each condition can contain multiple key-value pairs. The following figure illustrates this. Unless otherwise specified, all keys can have multiple values.



Let's say you want to let John use a resource only if a numeric value *foo* equals either A or B, and another numeric value *bar* equals C. You would create a condition block that looks like the following figure.

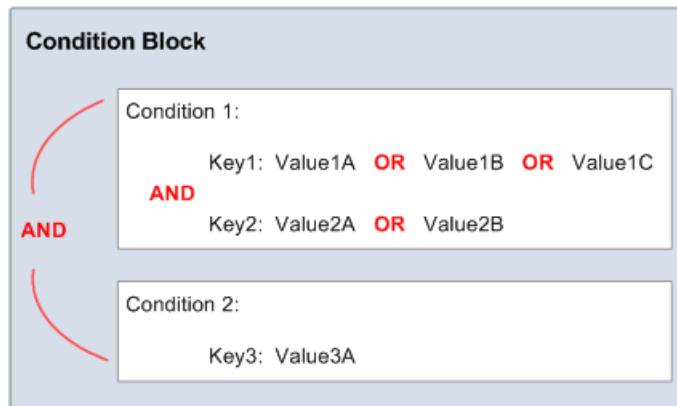


Let's say you also want to restrict John's access to after January 1, 2009. You would add another condition, DateGreaterThan, with a date equal to January 1, 2009. The condition block would then look like the following figure.



If there are multiple condition operators, or if there are multiple keys attached to a single condition operator, the conditions are evaluated using a logical AND. If a single condition operator includes multiple values for one key, that condition operator is evaluated using a logical OR. All condition

operators must be met for an allow or an explicit deny decision. If any one condition operator isn't met, the result is a deny.



As noted, AWS has predefined condition operators and keys (like `aws:currentTime`). Individual AWS services also define service-specific keys.

As an example, let's say you want to let user John access your Amazon SQS queue under the following conditions:

- The time is after 12:00 noon on 8/16/2013
 - The time is before 3:00 p.m. on 8/16/2013
 - The request (IAM or SQS) or message (SNS) comes from an IP address within the range 192.0.2.0 to 192.0.2.255 or 203.0.113.0 to 203.0.113.255.

Your condition block has three separate condition operators, and all three of them must be met for John to have access to your queue, topic, or resource.

The following shows what the condition block looks like in your policy. The two values for `aws:SourceIp` are evaluated using OR. The three separate condition operators are evaluated using AND.

```
"Condition" : {
    "DateGreaterThan" : {
        "aws:CurrentTime" : "2013-08-16T12:00:00Z"
    },
    "DateLessThan": {
        "aws:CurrentTime" : "2013-08-16T15:00:00Z"
    },
    "IpAddress" : {
        "aws:SourceIp" : ["192.0.2.0/24", "203.0.113.0/24"]
    }
}
```

Finally, under some circumstances, individual keys in a policy can contain multiple values, and you can use *condition set operators* to test these multi-valued keys against one or more values listed in the policy. For more information, see [Creating a Condition That Tests Multiple Key Values \(Set Operations\) \(p. 447\)](#).

IAM JSON Policy Elements: Condition Operators

Condition operators are the "verbs" of conditions and specify the type of comparison that IAM performs. The condition operators can be grouped into the following categories:

- [String \(p. 441\)](#)
- [Numeric \(p. 442\)](#)
- [Date and time \(p. 443\)](#)
- [Boolean \(p. 443\)](#)
- [Binary \(p. 444\)](#)
- [IP address \(p. 444\)](#)
- [Amazon Resource Name \(ARN\) \(p. 445\)](#) (available for only some services.)
- [...IfExists \(p. 446\)](#) (checks if the key value exists as part of another check)
- [Null check \(p. 447\)](#) (checks if the key value exists as a standalone check)

String Condition Operators

String condition operators let you construct `Condition` elements that restrict access based on comparing a key to a string value.

Condition Operator	Description
<code>StringEquals</code>	Exact matching, case sensitive
<code>StringNotEquals</code>	Negated matching
<code>StringEqualsIgnoreCase</code>	Exact matching, ignoring case
<code>StringNotEqualsIgnoreCase</code>	Negated matching, ignoring case
<code>StringLike</code>	Case-sensitive matching. The values can include a multi-character match wildcard (*) or a single-character match wildcard (?) anywhere in the string. Note If a key contains multiple values, <code>StringLike</code> can be qualified with set operators <code>—ForAllValues:StringLike</code> and <code>ForAnyValue:StringLike</code> . For more information, see Creating a Condition That Tests Multiple Key Values (Set Operations) (p. 447) .
<code>StringNotLike</code>	Negated case-sensitive matching. The values can include a multi-character match wildcard (*) or a single-character match wildcard (?) anywhere in the string.

For example, the following statement contains a `Condition` element that uses the `StringEquals` condition operator with the `aws:UserAgent` key to specify that the request must include a specific value in its user agent header.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "iam:*AccessKey*",
        "Resource": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:user/*",
        "Condition": {"StringEquals": {"aws:UserAgent": "Example Corp Java Client"}}
    }
}
```

The following example uses the `StringLike` condition operator to perform string matching with a [policy variable \(p. 451\)](#) to create a policy that lets an IAM user use the Amazon S3 console to manage his or her own "home directory" in an Amazon S3 bucket. The policy allows the specified actions on an S3 bucket as long as the `s3:prefix` matches any one of the specified patterns.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3>ListAllMyBuckets",
        "s3>GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::*
    },
    {
      "Effect": "Allow",
      "Action": "s3>ListBucket",
      "Resource": "arn:aws:s3:::BUCKET-NAME",
      "Condition": {"StringLike": {"s3:prefix": [
          "",
          "home/",
          "home/${aws:username}/"
        ]}}
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}",
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}/*"
      ]
    }
  ]
}
```

For an example of a policy that shows how to use the `Condition` element to restrict access to resources based on an application ID and a user ID for web identity federation, see [Amazon S3: Allows Amazon Cognito Users to Access Objects in Their Bucket \(p. 313\)](#).

Numeric Condition Operators

Numeric condition operators let you construct `Condition` elements that restrict access based on comparing a key to an integer or decimal value.

Condition Operator	Description
<code>NumericEquals</code>	Matching
<code>NumericNotEquals</code>	Negated matching
<code>NumericLessThan</code>	"Less than" matching
<code>NumericLessThanEquals</code>	"Less than or equals" matching
<code>NumericGreater Than</code>	"Greater than" matching
<code>NumericGreater ThanEquals</code>	"Greater than or equals" matching

For example, the following statement contains a Condition element that uses the NumericLessThanEquals condition operator with the s3:max-keys key to specify that the requester can list up to 10 objects in example_bucket at a time.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "s3>ListBucket",
        "Resource": "arn:aws:s3:::example_bucket",
        "Condition": {"NumericLessThanEquals": {"s3:max-keys": "10"}}
    }
}
```

Date Condition Operators

Date condition operators let you construct Condition elements that restrict access based on comparing a key to a date/time value. You use these condition operators with the aws:CurrentTime key or aws:EpochTime keys. You must specify date/time values with one of the [W3C implementations of the ISO 8601 date formats](#) or in epoch (UNIX) time.

Note

Wildcards are not permitted for date condition operators.

Condition Operator	Description
DateEquals	Matching a specific date
DateNotEquals	Negated matching
DateLessThan	Matching before a specific date and time
DateLessThanEquals	Matching at or before a specific date and time
DateGreaterThan	Matching after a specific date and time
DateGreaterThanOrEqual	Matching at or after a specific date and time

For example, the following statement contains a Condition element that uses the DateLessThan condition operator with the aws:CurrentTime key to specify that the request must be received before June 30, 2013.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "iam:*AccessKey*",
        "Resource": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:user/*",
        "Condition": {"DateLessThan": {"aws:CurrentTime": "2013-06-30T00:00:00Z"}}
    }
}
```

Boolean Condition Operators

Boolean conditions let you construct Condition elements that restrict access based on comparing a key to "true" or "false."

Condition Operator	Description
Bool	Boolean matching

For example, the following statement uses the `Bool` condition operator with the `aws:SecureTransport` key to specify that the request must use SSL.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
    "Resource": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:user/*",
    "Condition": {"Bool": {"aws:SecureTransport": "true"}}
  }
}
```

Binary Condition Operators

The `BinaryEquals` condition operator let you construct `Condition` elements that test key values that are in binary format. It compares the value of the specified key byte for byte against a [base-64](#) encoded representation of the binary value in the policy.

```
"Condition" : {
  "BinaryEquals": {
    "key" : "QmluYXJ5VmFsdWVJbkJhc2U2NA=="
  }
}
```

IP Address Condition Operators

IP address condition operators let you construct `Condition` elements that restrict access based on comparing a key to an IPv4 or IPv6 address or range of IP addresses. You use these with the `aws:SourceIp` key. The value must be in the standard CIDR format (for example, `203.0.113.0/24` or `2001:DB8:1234:5678::/64`). If you specify an IP address without the associated routing prefix, IAM uses the default prefix value of `/32`.

Some AWS services support IPv6, using `::` to represent a range of 0s. To learn whether a service supports IPv6, see the documentation for that service.

Condition Operator	Description
<code>IpAddress</code>	The specified IP address or range
<code>NotIpAddress</code>	All IP addresses except the specified IP address or range

For example, the following statement uses the `IpAddress` condition operator with the `aws:SourceIp` key to specify that the request must come from the IP range `203.0.113.0` to `203.0.113.255`.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
    "Resource": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:user/*",
    "Condition": {"IpAddress": {"aws:SourceIp": "203.0.113.0/24"}}
  }
}
```

```
    }
}
```

The `aws:SourceIp` condition key resolves to the IP address that the request originates from. If the requests originates from an Amazon EC2 instance, `aws:SourceIp` evaluates to the instance's public IP address.

The following example shows how to mix IPv4 and IPv6 addresses to cover all of your organization's valid IP addresses. We recommend that you augment your organization's policies with your IPv6 address ranges in addition to IPv4 ranges you already have to ensure the policies continue to work as you make the transition to IPv6.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "someservice:*",
    "Resource": "*",
    "Condition": {
      "IpAddress": {
        "aws:SourceIp": [
          "203.0.113.0/24",
          "2001:DB8:1234:5678::/64"
        ]
      }
    }
  }
}
```

The `aws:SourceIp` condition key works only in a JSON policy if you are calling the tested API directly as a user. If you instead use a service to call the target service on your behalf, the target service sees the IP address of the calling service rather than the IP address of the originating user. This can happen, for example, if you use AWS CloudFormation to call Amazon EC2 to construct instances for you. There is currently no way to pass the originating IP address through a calling service to the target service for evaluation in a JSON policy. For these types of service API calls, do not use the `aws:SourceIp` condition key.

[Amazon Resource Name \(ARN\) Condition Operators](#)

Amazon Resource Name (ARN) condition operators let you construct Condition elements that restrict access based on comparing a key to an ARN. The ARN is considered a string. This value is available for only some services; not all services support request values that can be compared as ARNs.

Condition Operator	Description
<code>ArnEquals</code> , <code>ArnLike</code>	Case-sensitive matching of the ARN. Each of the six colon-delimited components of the ARN is checked separately and each can include a multi-character match wildcard (*) or a single-character match wildcard (?). These behave identically.
<code>ArnNotEquals</code> , <code>ArnNotLike</code>	Negated matching for ARN. These behave identically.

The following example shows a policy you need to attach to any Amazon SNS queue that you want to send SNS messages to. It gives Amazon SNS permission to send messages to the queue (or queues) of your choice, but only if the service is sending the messages on behalf of a particular Amazon SNS topic (or topics). You specify the queue in the `Resource` field, and the Amazon SNS topic as the value for the `SourceArn` key.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"AWS": "123456789012"},
            "Action": "SQS:SendMessage",
            "Resource": "arn:aws:sqs:REGION:123456789012:QUEUE-ID",
            "Condition": {"ArnEquals": {"aws:SourceArn": "arn:aws:sns:REGION:123456789012:TOPIC-ID"}}
        }
    ]
}
```

...IfExists Condition Operators

You can add `IfExists` to the end of any condition operator name except the `Null` condition—for example, `StringLikeIfExists`. You do this to say "If the policy key is present in the context of the request, process the key as specified in the policy. If the key is not present, the condition evaluate the condition element as true." Other condition elements in the statement can still result in a nonmatch, but not a missing key when checked with `...IfExists`.

Example using `IfExists`

Many condition keys describe information about a certain type of resource and only exist when accessing that type of resource. These condition keys are not present on other types of resources. This doesn't cause an issue when the policy statement applies to only one type of resource. However, there are cases where a single statement can apply to multiple types of resources, such as when the policy statement references actions from multiple services or when a given action within a service accesses several different resource types within the same service. In such cases, including a condition key that applies to only one of the resources in the policy statement can cause the `Condition` element in the policy statement to fail such that the statement's "Effect" does not apply.

For example, consider the following policy example:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "THISPOLICYDOESNOTWORK",
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "*",
            "Condition": {"StringLike": {"ec2:InstanceType": [
                "t1.*",
                "t2.*",
                "m3.*"
            ]}}
        }
    ]
}
```

The *intent* of the preceding policy is to enable the user to launch any instance that is type t1, t2 or m3. However, launching an instance actually requires accessing many resources in addition to the instance itself; for example, images, key pairs, security groups, etc. The entire statement is evaluated against every resource that is required to launch the instance. These additional resources do not have the `ec2:InstanceType` condition key, so the `StringLike` check fails, and the user is not granted the ability to launch *any* instance type. To address this, use the `StringLikeIfExists` condition operator instead. This way, the test only happens if the condition key exists. You could read the following as: "If the resource being checked has an `"ec2:InstanceType"` condition key, then allow the action only if the key value begins with `"t1.*"`, `"t2.*"`, or `"m3.*"`. If the resource being checked does not have that condition key, then don't worry about it."

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "*",
            "Condition": {"StringLikeIfExists": {"ec2:InstanceType": [
                "t1.*",
                "t2.*",
                "m3.*"
            ]}}}
    ]
}
```

Condition Operator to Check Existence of Condition Keys

Use a `Null` condition operator to check if a condition key is present at the time of authorization. In the policy statement, use either `true` (the key doesn't exist — it is null) or `false` (the key exists and its value is not null).

For example, you can use this condition operator to determine whether a user is using their own credentials for the operation or temporary credentials. If the user is using temporary credentials, then the key `aws:TokenIssueTime` exists and has a value. The following example shows a condition that states that the user must not be using temporary credentials (the key must not exist) for the user to use the Amazon EC2 API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "ec2:*",
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {"Null": {"aws:TokenIssueTime": "true"}}
        }
    ]
}
```

Creating a Condition That Tests Multiple Key Values (Set Operations)

This topic discusses how to create a policy `Condition` element that lets you test multiple values for a single key in a request. In effect, you can create a condition that uses set operations. This type of condition is useful for creating policies that allow fine-grained control for DynamoDB tables (for example, allowing or denying access to particular attributes).

To create this type of condition, you can use the `ForAllValues` or `ForAnyValue` qualifier with the condition operator. These qualifiers add set-operation functionality to the condition operator so that you can test multiple request values against multiple condition values.

Contents

- [Introduction \(p. 447\)](#)
- [Examples of Condition Set Operators \(p. 448\)](#)
- [Evaluation Logic for Condition Set Operators \(p. 449\)](#)

Introduction

In some situations, you need to create a policy in which you test multiple values in a request against one or more values that you specify in the policy. A good example is a request for attributes from an Amazon DynamoDB table. Imagine an Amazon DynamoDB table named `Thread` that is used to store information about threads in a technical support forum. The table might have attributes like `ID`, `UserName`, `PostDateTime`, `Message`, and `Tags` (among others).

```
{
  ID=101
  UserName=Bob
  PostDateTime=20130930T231548Z
  Message="A good resource for this question is http://aws.amazon.com/documentation/"
  Tags=[ "AWS", "Database", "Security" ]
}
```

You might want to create a policy that allows users to see only some attributes—for example, maybe you want to let these users to see only `PostDateTime`, `Message`, and `Tags`. If the user's request contains any of these attributes, it is allowed; but if the request contains any other attributes (for example, `ID`), the request is denied. Logically speaking, you want to create a list of allowed attributes (`PostDateTime`, `Message`, `Tags`) and indicate in the policy that all of the user's requested attributes must be in that list of allowed attributes.

Or you might want to make sure that users are explicitly forbidden to include some attributes in a request, such as the `ID` and `UserName` attributes. For example, you might exclude attributes when the user is updating the DynamoDB table, because an update (`PUT` operation) should not change certain attributes. In that case, you create a list of forbidden attributes (`ID`, `UserName`), and if any of the user's requested attributes match any of the forbidden attributes, the request is denied.

To support these scenarios, you can use the following modifiers to a condition operator:

- `ForAnyValue` – The condition returns true if any one of the key values in the request matches any one of the condition values in the policy.
- `ForAllValues` – The condition returns true if there's a match between every one of the specified key values in the request and at least one value in the policy.

For information about how set operators are used in DynamoDB to implement fine-grained access to individual data items and attributes, see [Fine-Grained Access Control for DynamoDB](#) in the *Amazon DynamoDB Developer Guide* guide.

Examples of Condition Set Operators

The following example policy shows how to use the `ForAllValues` qualifier with the `StringLike` condition operator. The condition allows a user to request *only* the attributes `PostDateTime`, `Message`, or `Tags` from the DynamoDB table named `Thread`.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "GetItem",
    "Resource": "arn:aws:dynamodb:REGION:ACCOUNT-ID-WITHOUT-HYPHENS:table/Thread",
    "Condition": {"ForAllValues:StringLike": {"dynamodb:requestedAttributes": [
      "PostDateTime",
      "Message",
      "Tags"
    ]}}
  }]
}
```

If the user makes a request to DynamoDB to get the attributes `PostDateTime` and `Message` from the `Threads` table, the request is allowed, because the user's requested attributes all match values specified in the policy. However, if the user's request includes the `ID` attribute, the request fails, because `ID` is not within the list of allowed attributes, and the `ForAllValues` qualifier requires all requested values to be listed in the policy.

The following example shows how to use the `ForAnyValue` qualifier as part of a policy that denies access to the `ID` and `PostDateTime` attributes if the user tries to perform the `PutItem` action—that is, if the user tries to update either of those attributes. Notice that the `Effect` element is set to `Deny`.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "Action": "PutItem",
    "Resource": "arn:aws:dynamodb:REGION:ACCOUNT-ID-WITHOUT-HYPHENS:table/Thread",
    "Condition": {"ForAnyValue:StringLike": {"dynamodb:requestedAttributes": [
      "ID",
      "PostDateTime"
    ]}}
  }]
}
```

Imagine that the user makes a request to update the `PostDateTime` and `Message` attributes. The `ForAnyValue` qualifier determines whether any of the requested attributes appear in the list in the policy. In this case, there is one match (`PostDateTime`), so the condition is true. Assuming the other values in the request also match (for example, the resource), the overall policy evaluation returns true. Because the policy's effect is `Deny`, the request is denied.

Imagine the user instead makes a request to perform `PutItem` with just the `UserName` attribute. None of the attributes in the request (just `UserName`) match any of attributes listed in the policy (`ID`, `PostDateTime`). The condition returns false, so the effect of the policy (`Deny`) is also false, and the request is not denied by this policy. (For the request to succeed, it must be explicitly allowed by a different policy. It is not explicitly denied by this policy, but all requests are implicitly denied.)

Evaluation Logic for Condition Set Operators

This section discusses the specifics of the evaluation logic used with the `ForAllValues` and `ForAnyValue` qualifiers. The following table illustrates possible keys that might be included in a request (`PostDateTime` and `UserName`) and a policy condition that includes the values `PostDateTime`, `Message`, and `Tags`.

Key (in the request)	Condition Value (in the policy)
<code>PostDateTime</code>	<code>PostDateTime</code>
<code>UserName</code>	<code>Message</code>
	<code>Tags</code>

The evaluation for the combination is this:

<code>PostDateTime</code> matches <code>PostDateTime</code> ?
<code>PostDateTime</code> matches <code>Message</code> ?
<code>PostDateTime</code> matches <code>Tags</code> ?
<code>UserName</code> matches <code>PostDateTime</code> ?
<code>UserName</code> matches <code>Message</code> ?
<code>UserName</code> matches <code>Tags</code> ?

The result of the condition operator depends on which modifier is used with the policy condition:

- **ForAllValues.** If every key in the request (`PostDateTime` or `UserName`) matches at least one condition value in the policy (`PostDateTime`, `Message`, `Tags`), the condition operator returns true. Stated another way, in order for the condition to be true, (`PostDateTime` must equal `PostDateTime`, `Message`, or `Tags`) *and* (`UserName` must equal `PostDateTime`, `Message`, or `Tags`).
- **ForAnyValue.** If any combination of request value and policy value (any one of the six in the example) returns true, the condition operator returns true.

The following policy includes a `ForAllValues` qualifier:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "GetItem",
      "Resource": "arn:aws:dynamodb:REGION:ACCOUNT-ID-WITHOUT-HYPHENS:table/Thread",
      "Condition": {"ForAllValues:StringLike": {"dynamodb:requestedAttributes": [
          "PostDateTime",
          "Message",
          "Tags"
        ]}}
    }
  ]
}
```

Suppose that the user makes a request to DynamoDB to get the attributes `PostDateTime` and `UserName`. The keys in the request and condition values in the policy are these:

Key (in the request)	Condition Value (in the policy)
<code>PostDateTime</code>	<code>PostDateTime</code>
<code>UserName</code>	<code>Message</code>
	<code>Tags</code>

The evaluation for the combination is this:

<code>PostDateTime</code> matches <code>PostDateTime</code> ?	<code>True</code>
<code>PostDateTime</code> matches <code>Message</code> ?	<code>False</code>
<code>PostDateTime</code> matches <code>Tags</code> ?	<code>False</code>
<code>UserName</code> matches <code>PostDateTime</code> ?	<code>False</code>
<code>UserName</code> matches <code>Message</code> ?	<code>False</code>
<code>UserName</code> matches <code>Tags</code> ?	<code>False</code>

The policy includes the `ForAllValues` condition operator modifier, meaning that there must be at least one match for `PostDateTime` and one match for `UserName`. There's no match for `UserName`, so the condition operator returns false, and the policy does not allow the request.

The following policy includes a `ForAnyValue` qualifier:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "PutItem",
      "Resource": "arn:aws:dynamodb:REGION:ACCOUNT-ID-WITHOUT-HYPHENS:table/Thread",
      "Condition": {"ForAnyValue:StringLike": {"dynamodb:requestedAttributes": [
        "ID",
        "PostDateTime"
      ]}}
    }
  ]
}
```

Notice that the policy includes "Effect": "Deny" and the action is PutItem. Imagine that the user makes a PutItem request that includes the attributes UserName, Message, and PostDateTime. The evaluation is this:

UserName matches ID?	False
UserName matches PostDateTime?	False
Messages matches ID?	False
Message matches PostDateTime?	False
PostDateTime matches ID?	False
PostDateTime matches PostDateTime?	True

With the modifier ForAnyValue, if any one of these tests returns true, the condition returns true. The last test returns true, so the condition is true; because the Effect element is set to Deny, the request is denied.

Note

If the key values in the request resolve to an empty data set (for example, an empty string), a condition operator modified by ForAllValues returns true, and a condition operator modified by ForAnyValue returns false.

IAM Policy Elements: Variables

Topics

- [Introduction \(p. 451\)](#)
- [Where You Can Use Policy Variables \(p. 453\)](#)
- [Request Information That You Can Use for Policy Variables \(p. 454\)](#)
- [For More Information \(p. 457\)](#)

Introduction

In IAM policies, many actions allow you to provide a name for the specific resources that want to control access to. For example, the following policy allows the user to list, read, and write objects with a prefix David in the Amazon S3 bucket mybucket.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["s3>ListBucket"],
      ...
    }
  ]
}
```

```

        "Effect": "Allow",
        "Resource": ["arn:aws:s3:::mybucket"],
        "Condition": {"StringLike": {"s3:prefix": ["David/*"]}}
    },
    {
        "Action": [
            "s3:GetObject",
            "s3:PutObject"
        ],
        "Effect": "Allow",
        "Resource": ["arn:aws:s3:::mybucket/David/*"]
    }
]
}

```

In some cases, you might not know the exact name of the resource when you write the policy. You might want to generalize the policy so it works for many users without having to make a unique copy of the policy for each user. For example, consider writing a policy allow each user to have access to his or her own objects in an Amazon S3 bucket, as in the previous example. However, instead of creating a separate policy for each user that explicitly specifies the user's name as part of the resource, you want to create a single group policy that works for any user in that group.

You can do this by using *policy variables*, a feature that lets you specify placeholders in a policy. When the policy is evaluated, the policy variables are replaced with values that come from the context of the request itself.

The following example shows a policy for an Amazon S3 bucket that uses a policy variable.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": ["s3>ListBucket"],
            "Effect": "Allow",
            "Resource": ["arn:aws:s3:::mybucket"],
            "Condition": {"StringLike": {"s3:prefix": ["${aws:username}/*"]}}
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": ["arn:aws:s3:::mybucket/${aws:username}/*"]
        }
    ]
}

```

When this policy is evaluated, IAM replaces the variable \${aws:username} with the [friendly name \(p. 410\)](#) of the actual current user. This means that a single policy applied to a group of users can control access to a bucket by using the user name as part of the resource's name.

The variable is marked using a \$ prefix followed by a pair of curly braces ({ }). Inside the \${ } characters, you can include the name of the value from the request that you want to use in the policy. The values you can use are discussed later on this page.

Note

In order to use policy variables, you must include the `Version` element in a statement, and the version must be set to a version that supports policy variables. Variables were introduced in version 2012-10-17. Earlier versions of the policy language don't support policy variables. If you don't include the `Version` element and set it to an appropriate version date, variables like \${aws:username} are treated as literal strings in the policy.

A **Version** policy element is different from a policy version. The **Version** policy element is used within a policy and defines the version of the policy language. A policy version, on the other hand, is created when you make changes to a customer managed policy in IAM. The changed policy doesn't overwrite the existing policy. Instead, IAM creates a new version of the managed policy. To learn more about the **Version** policy element see [the section called "Version" \(p. 426\)](#). To learn more about policy versions, see [the section called "Versioning IAM Policies" \(p. 335\)](#).

You can use policy variables in a similar way to allow each user to manage his or her own access keys. A policy that allows a user to programmatically change the access key for user David looks like this:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["iam:*AccessKey*"],
      "Effect": "Allow",
      "Resource": ["arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:user/David"]
    }
  ]
}
```

If this policy is attached to user David, that user can change his own access key. As with the policies for accessing user-specific Amazon S3 objects, you would have to create a separate policy for each user that includes the user's name. You would then attach each policy to the individual users.

By using a policy variable, you can create a policy like this:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["iam:*AccessKey*"],
      "Effect": "Allow",
      "Resource": ["arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:user/${aws:username}"]
    }
  ]
}
```

When you use a policy variable for the user name like this, you don't have to have a separate policy for each individual user. Instead, you can attach this new policy to an IAM group that includes everyone who should be allowed to manage their own access keys. When a user makes a request to modify his or her access key, IAM substitutes the user name from the current request for the \${aws:username} variable and evaluates the policy.

Where You Can Use Policy Variables

You can use policy variables in the **Resource** element and in string comparisons in the **Condition** element.

Resource Element

A policy variable can appear as the last part of the [ARN](#) that identifies a resource. The following policy might be attached to a group. It gives each of the users in the group full programmatic access to a user-specific object (their own "home directory") in Amazon S3.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["s3>ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket"],
      "Condition": {"StringLike": {"s3:prefix": ["${aws:username}/*"]}}
    }
  ]
}
```

```

    },
    {
        "Action": [
            "s3:GetObject",
            "s3:PutObject"
        ],
        "Effect": "Allow",
        "Resource": ["arn:aws:s3:::mybucket/${aws:username}/*"]
    }
]
}

```

Note

This example uses the `aws:username` key, which returns the user's friendly name (like "Adele" or "David"). Under some circumstances, you might want to use the `aws:userid` key instead, which is a globally unique value. For more information, see [Unique IDs \(p. 413\)](#).

The following policy might be used for an IAM group. It gives users in that group the ability to create, use, and delete queues that have their names and that are in the us-east-2 region.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sns:*",
            "Resource": "arn:aws:sns:us-east-2:*:${aws:username}-queue"
        }
    ]
}
```

Condition Element

A policy variable can also be used for Condition values in any condition that involves the string operators (`StringEquals`, `StringLike`, `StringNotLike`, etc.) or the ARN operators (`ArnEquals`, `ArnLike`, etc.). The following Amazon SNS topic policy gives users in AWS account 999999999999 the ability to manage (perform all actions for) the topic only if the URL matches their AWS user name.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Principal": {"AWS": "999999999999"},
            "Effect": "Allow",
            "Action": "sns:*",
            "Condition": {"StringLike": {"sns:endpoint": "https://example.com/${aws:username}/*"}}
        }
    ]
}
```

Request Information That You Can Use for Policy Variables

The values that can be substituted for policy variables must come from the current request context ([p. 459](#)).

Information Available in All Requests

Policies contain keys whose values you can use as policy variables. (Under some circumstances, the keys do not contain a value—see the information that follows this list.)

- **`aws:CurrentTime`** This can be used for conditions that check the date and time.
- **`aws:EpochTime`** This is the date in epoch or UNIX time, for use with date/time conditions.
- **`aws:TokenIssueTime`** This is the date and time that temporary security credentials were issued and can be used with date/time conditions. **Note:** This key is only available in requests that are signed

using temporary security credentials. For more information about temporary security credentials, see [Temporary Security Credentials \(p. 231\)](#).

- **aws:principaltype** This value indicates whether the principal is an account, user, federated, or assumed role—see the explanation that follows later.
- **aws:SecureTransport** This is a Boolean value that represents whether the request was sent using SSL.
- **aws:SourceIp** This is the requester's IP address, for use with IP address conditions. Refer to [IP Address Condition Operators \(p. 444\)](#) for information about when SourceIp is valid and when you should use a VPC-specific key instead.
- **aws:UserAgent** This value is a string that contains information about the requester's client application.
- **aws:userid** This value is the unique ID for the current user—see the chart that follows.
- **aws:username** This is a string containing the [friendly name \(p. 410\)](#) of the current user—see the chart that follows.
- **ec2:SourceInstanceARN** This is the Amazon Resource Name (ARN) of the Amazon EC2 instance from which the request is made. This key is present only when the request comes from an Amazon EC2 instance using an IAM role associated with an EC2 instance profile.

Important

Key names are case-insensitive. For example, `aws:CurrentTime` is equivalent to `AWS:currenttime`.

The values for `aws:username`, `aws:userid`, and `aws:principaltype` depend on what type of principal initiated the request—whether the request was made using the credentials of an AWS account, an IAM user, an IAM role, and so on. The following list shows values for these keys for different types of principal.

- **AWS Account**
 - `aws:username`: (not present)
 - `aws:userid`: AWS account ID
 - `aws:principaltype`: Account
- **IAM user**
 - `aws:username`: *IAM-user-name*
 - `aws:userid`: [unique ID \(p. 413\)](#)
 - `aws:principaltype`: User
- **Federated user**
 - `aws:username`: (not present)
 - `aws:userid`: *account:caller-specified-name*
 - `aws:principaltype`: FederatedUser
- **Web federated user and SAML federated user**

Note

For information about policy keys that are available when you use web identity federation, see [Identifying Users with Web Identity Federation \(p. 147\)](#).

- `aws:username`: (not present)
- `aws:userid`: (not present)
- `aws:principaltype`: AssumedRole
- **Assumed role**
 - `aws:username`: (not present)
 - `aws:userid`: *role-id:caller-specified-role-name*
 - `aws:principaltype`: Assumed role

- **Role assigned to Amazon EC2 instance**
 - aws:username: (not present)
 - aws:userid: *role-id:ec2-instance-id*
 - aws:principaltype: Assumed role
- **Anonymous caller** (Amazon SQS Amazon SNS and Amazon S3 only)
 - aws:username: (not present)
 - aws:userid: (not present)
 - aws:principaltype: Anonymous

In this list:

- *not present* means that the value is not in the current request information, and any attempt to match it fails and causes the request to be denied.
- *role-id* is a unique identifier assigned to each role at creation. You can display the role ID with the AWS CLI command: aws iam get-role --role-name *rolename*
- *caller-specified-name* and *caller-specified-role-name* are names that are passed by the calling process (such as an application or service) when it makes a call to get temporary credentials.
- *ec2-instance-id* is a value assigned to the instance when it is launched and appears on the **Instances** page of the Amazon EC2 console. You can also display the instance ID by running the AWS CLI command: aws ec2 describe-instances

Information Available in Requests for Federated Users

Federated users are users who are authenticated using a system other than IAM. For example, a company might have an application for use in-house that makes calls to AWS. It might be impractical to give an IAM identity to every corporate user who uses the application. Instead, the company might use a proxy (middle-tier) application that has a single IAM identity, or the company might use a SAML identity provider (IdP). The proxy application or SAML IdP authenticates individual users using the corporate network. A proxy application can then use its IAM identity to get temporary security credentials for individual users. A SAML IdP can in effect exchange identity information for AWS temporary security credentials. The temporary credentials can then be used to access AWS resources.

Similarly, you might create an app for a mobile device in which the app needs to access AWS resources. In that case, you might use *web identity federation*, where the app authenticates the user using a well-known identity provider like Login with Amazon, Amazon Cognito, Facebook, or Google. The app can then use the user's authentication information from these providers to get temporary security credentials for accessing AWS resources.

The recommended way to use web identity federation is by taking advantage of Amazon Cognito and the AWS mobile SDKs. For more information, see the following:

- [Amazon Cognito Overview](#) in the *AWS Mobile SDK for Android Developer Guide*
- [Amazon Cognito Overview](#) in the *AWS Mobile SDK for iOS Developer Guide*
- [Common Scenarios for Temporary Credentials \(p. 232\)](#).

Service-Specific Information

Requests can also include service-specific keys and values in its request context. Examples include the following:

- s3:prefix
- s3:max-keys

- `s3:x-amz-acl`
- `sns:Endpoint`
- `sns:Protocol`

For information about service-specific keys that you can use to get values for policy variables, refer to the documentation for the individual services. For example, see the following topics:

- [Bucket Keys in Amazon S3 Policies](#) in the *Amazon Simple Storage Service Developer Guide*.
- [Amazon SNS Keys](#) in the *Amazon Simple Notification Service Developer Guide*.

Special Characters

There are a few special predefined policy variables that have fixed values that enable you to represent characters that otherwise have special meaning. If these special characters are part of the string, you are trying to match and you inserted them literally they would be misinterpreted. For example, inserting an * asterisk in the string would be interpreted as a wildcard, matching any characters, instead of as a literal *. In these cases, you can use the following predefined policy variables:

- `${*}` - use where you need an * asterisk character.
- `${?}` - use where you need a ? question mark character.
- `${$}` - use where you need a \$ dollar sign character.

These predefined policy variables can be used in any string where you can use regular policy variables.

For More Information

For more information about policies, see the following:

- [IAM Policies \(p. 275\)](#)
- [Example Policies \(p. 295\)](#)
- [IAM JSON Policy Elements Reference \(p. 426\)](#)
- [IAM JSON Policy Evaluation Logic \(p. 458\)](#)
- [About Web Identity Federation \(p. 143\)](#)

IAM JSON Policy Elements: Supported Data Types

This section lists the data types that are supported when you specify values in JSON policies. The policy language doesn't support all types for each policy element; for information about each element, see the preceding sections.

- Strings
- Numbers (Ints and Floats)
- Boolean
- Null
- Lists
- Maps
- Structs (which are just nested Maps)

The following table maps each data type to the serialization. Note that all policies must be in UTF-8. For information about the JSON data types, go to [RFC 4627](#).

Type	JSON
String	String
Integer	Number
Float	Number
Boolean	true false
Null	null
Date	String adhering to the W3C Profile of ISO 8601
IpAddress	String adhering to RFC 4632
List	Array
Object	Object

IAM JSON Policy Evaluation Logic

Topics

- [Policy Evaluation Basics \(p. 458\)](#)
- [The Impact of AWS Organizations on IAM Policies \(p. 458\)](#)
- [The Request Context \(p. 459\)](#)
- [Determining Whether a Request is Allowed or Denied \(p. 459\)](#)
- [The Difference Between Denying by Default and Explicit Deny \(p. 462\)](#)

Policy Evaluation Basics

When an AWS service receives a request, the request is first authenticated using information about the access key ID and signature. (A few services, like Amazon S3, allow requests from anonymous users.) If the request passes authentication, AWS then determines whether the requester is authorized to perform the action represented by the request.

Requests that are made by the AWS account root user are allowed for resources in that account. However, if the request is made using the credentials of an IAM user, or if the request is signed using temporary credentials that are granted by AWS STS, AWS uses the permissions defined in one or more IAM policies to determine whether the user's request is authorized.

Note

Amazon S3 supports Access Control Lists (ACLs) and resource-level policies for buckets and objects. The permissions established using ACLs and bucket-level policies can affect what actions the root user is allowed to perform on a bucket. For more information, see [Guidelines for Using the Available Access Policy Options](#) in the *Amazon Simple Storage Service Developer Guide*.

The Impact of AWS Organizations on IAM Policies

AWS Organizations is a service that enables you to group together and centrally manage the 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. These SCPs serve as "filters" or "guardrails" that limit what services and actions can be accessed by the IAM users, groups, and roles in those accounts. If an SCP attached to an account denies access to a service, such as S3, then no user in that

account can access any S3 API, even if the user has administrator permissions in the account. Even the AWS account root user is denied access to S3 APIs.

In summary, an IAM user in an account that is in an organization can use the *intersection* of the permissions allowed by both Organizations SCPs and by the IAM permission policies attached to the user by the account's administrator. In other words, only those allowed by both IAM and Organizations.

For more information about Organizations and SCPs, see [About Service Control Policies](#) in the *AWS Organizations User Guide*.

The Request Context

When AWS authorizes a request, information about the request is assembled from several sources:

- Principal (the requester), which is determined based on the secret access key. This might represent the root user, an IAM user, a federated user (via STS), or an assumed role, and includes the aggregate permissions that are associated with that principal.
- Environment data, such as the IP address, user agent, SSL enabled, the time of day, etc. This information is determined from the request.
- Resource data, which pertains to information that is part of the resource being requested. This can include information such as a DynamoDB table name, a tag on an Amazon EC2 instance, etc.

This information is gathered into a *request context*, which is a collection of information that's derived from the request. During evaluation, AWS uses values from the request context to determine whether to allow or deny the request. For example, does the action in the request context match an action in the Action element? If not, the request is denied. Similarly, does the resource in the request context match one of the resources in the Resource element? If not, the request is denied.

This is also how the keys work that you can use in the Condition element. For example, for the following policy fragment, AWS uses the date and time from the current request context for the aws:CurrentTime key and then performs the DateGreaterThanOrEqual and DateLessThanOrEqual comparisons.

```
"Condition": {
    "DateGreaterThanOrEqual": {
        "aws:CurrentTime" : "2013-08-16T12:00:00Z"
    },
    "DateLessThanOrEqual": {
        "aws:CurrentTime" : "2013-08-16T15:00:00Z"
    }
}
```

Policy variables like \${aws:username} also work like this. In the following policy fragment, AWS gets the user name from the request context and uses it in the policy at the place where the \${aws:username} occurs.

```
"Resource": [
    "arn:aws:s3:::mybucket/${aws:username}/*"
]
```

Determining Whether a Request is Allowed or Denied

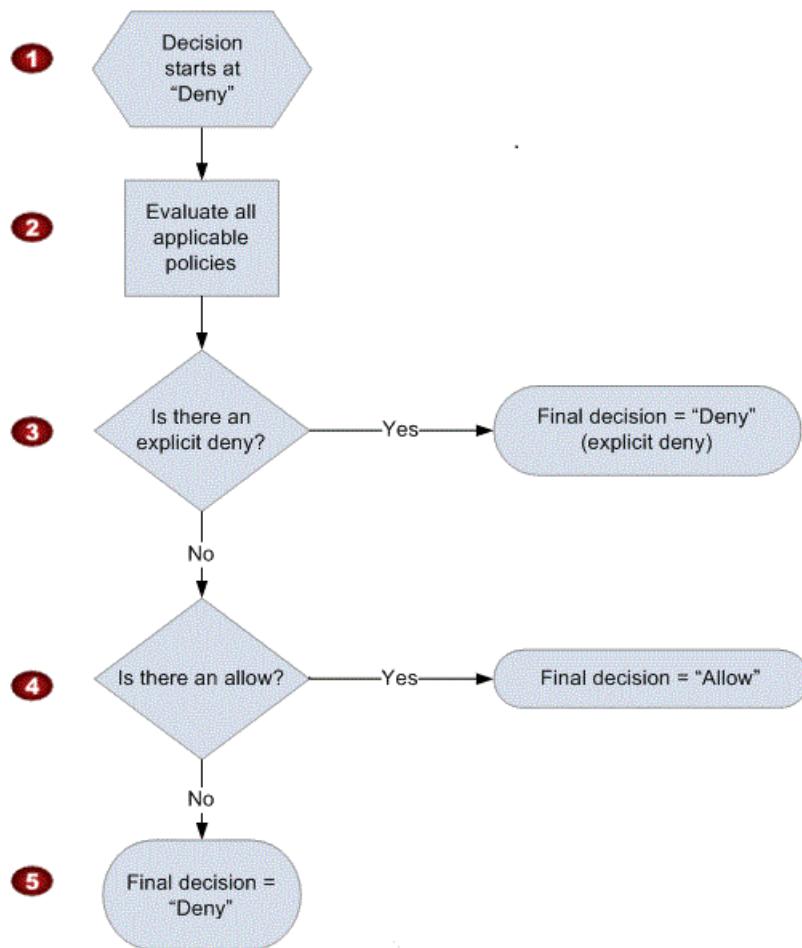
When a request is made, the AWS service decides whether a given request should be allowed or denied. The evaluation logic follows these rules:

- By default, all requests are denied. (In general, requests made using the account credentials for resources in the account are always allowed.)
- An explicit allow overrides this default.

- An explicit deny overrides any allows.

The order in which the policies are evaluated has no effect on the outcome of the evaluation. All policies are evaluated, and the result is always that the request is either allowed or denied.

The following flow chart provides details about how the decision is made.



1. The decision starts by assuming that the request will be denied.
2. The enforcement code evaluates all user-based and resource-based policies that are applicable to the request (based on the resource, principal, action, and conditions).

The order in which the enforcement code evaluates the policies is not important.

3. In all those policies, the enforcement code looks for an explicit deny instruction that would apply to the request.

If the code finds even one explicit deny that applies, the code returns a decision of Deny and the process is finished.

4. If no explicit deny is found, the code looks for any Allow instructions that would apply to the request.

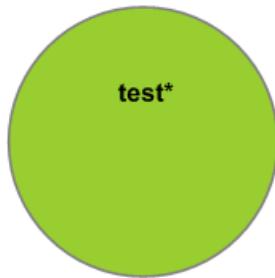
If it finds even one explicit allow, the code returns a decision of Allow and the service continues to process the request.

5. If no allow is found, the final decision is Deny.

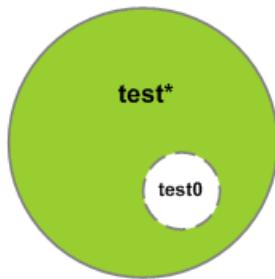
If the code encounters an error at any point during the evaluation, then it will generate an exception and close.

The following example illustrates how you can use an explicit deny to override a broad policy that allows access to a wide set of resources. Let's say that you give an IAM group permissions to use any Amazon SQS queues in your AWS account whose names begin with the string test.

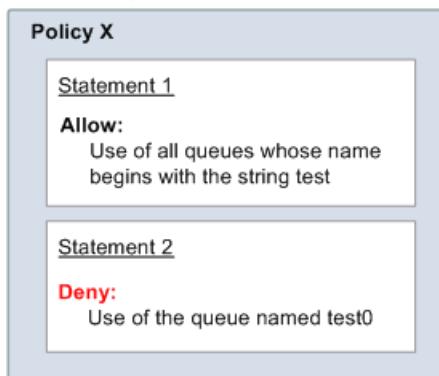
The following diagram represents that set of queues.



Let's say that you have a queue called test0 that you want to remove the group's access to. The following diagram represents that set of queues.



You could add another policy to the group, or another statement to the existing policy, that explicitly denies the group's access to the test0 queue. The group would still have access to all other queues whose names begin with the string test. The following diagram illustrates those two statements in the policy.



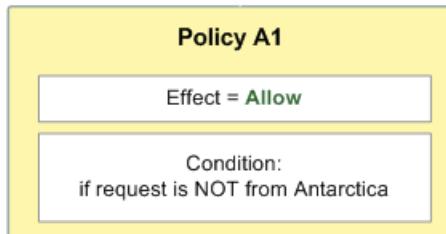
When any user in the group makes a request to use the queue `test0`, the explicit deny overrides the `allow`, and the user is denied access to the queue.

The Difference Between Denying by Default and Explicit Deny

A policy results in a deny if the policy doesn't directly apply to the request. For example, if a user makes a request to use Amazon SQS, but the only policy that applies to the user states that the user can use Amazon S3, the request is denied.

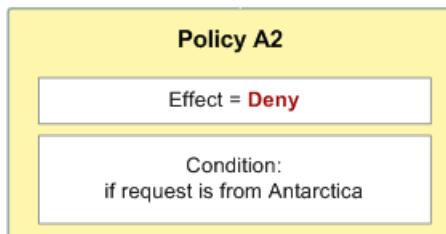
A policy also results in a deny if a condition in a statement isn't met. If all conditions in the statement are met, the policy results in either an allow or an explicit deny, based on the value of the `Effect` element in the policy. Policies don't specify what to do if a condition isn't met, so the result in that case is `Deny`.

For example, let's say you want to prevent requests that come from Antarctica. You write a policy called Policy A1 that allows a request only if it doesn't come from Antarctica. The following diagram illustrates the policy.



If someone sends a request from the U.S., the condition is met (the request is not from Antarctica). Therefore, the result is `Allow`. But if someone sends a request from Antarctica, the condition isn't met, and the policy's result is therefore `Deny`.

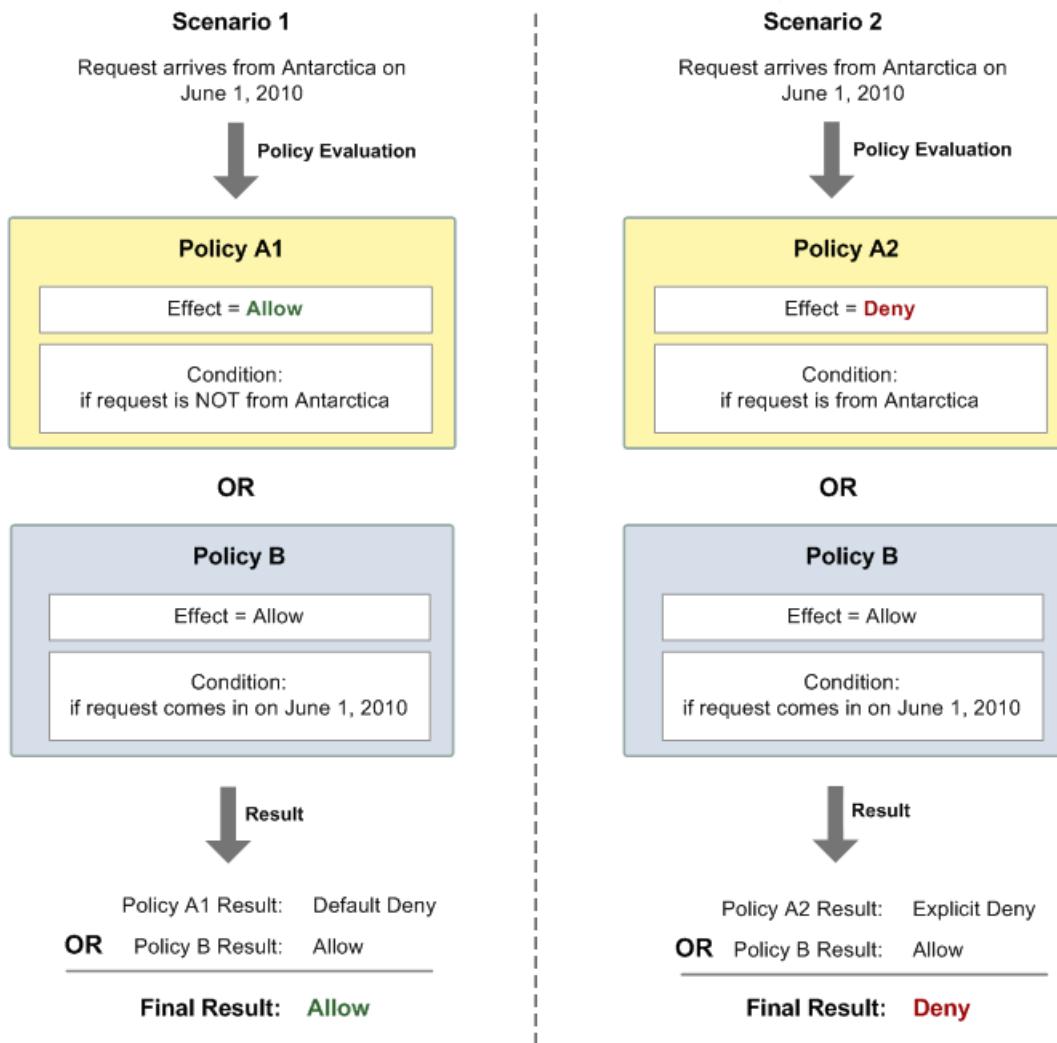
You could *explicitly* deny access from Antarctica by creating a different policy (named Policy A2) as in the following diagram.



If this policy applies to a user who sends a request from Antarctica, the condition is met, and the policy's result is therefore `Deny`.

The distinction between a request being denied by default and an explicit deny in a policy is important. By default, a request is denied, but this can be overridden by an allow. In contrast, if a policy explicitly denies a request, that deny can't be overridden.

For example, let's say there's another policy that allows requests if they arrive on June 1, 2010. How does this policy affect the outcome when evaluated together with the policy that restricts access from Antarctica? We'll compare the outcome when evaluating the date-based policy (we'll call it Policy B) with the preceding policies (A1 and A2). Scenario 1 evaluates Policy A1 with Policy B, and Scenario 2 evaluates Policy A2 with Policy B. The following figure show the results when a request comes in from Antarctica on June 1, 2010.



In Scenario 1, Policy A1 returns Deny because requests are allowed only if they don't come from Antarctica; any other condition (requests that do come from Antarctica) are denied by default. Policy B returns an allow because the request arrives on June 1, 2010. The allow from Policy B overrides the default deny from Policy A1, and the request is therefore allowed.

In Scenario 2, Policy A2 returns an explicit deny, as described earlier in this section. Again, Policy B returns an allow. However, the explicit deny from Policy A2 overrides the allow from Policy B, and the request is therefore Deny.

Grammar of the IAM JSON Policy Language

This page presents a formal grammar for the language used to create JSON policies in IAM. We present this grammar so that you can understand how to construct and validate policies.

For examples of policies, see the following topics:

- [IAM Policies \(p. 275\)](#)
- [Example Policies \(p. 295\)](#)

- [Example Policies for Working in the Amazon EC2 Console](#) and [Example Policies for Working With the AWS CLI, the Amazon EC2 CLI, or an AWS SDK](#) in the *Amazon EC2 User Guide for Linux Instances*.
- [Bucket Policy Examples](#) and [User Policy Examples](#) in the *Amazon Simple Storage Service Developer Guide*.

For examples of policies used in other AWS services, go to the documentation for those services.

Topics

- [The Policy Language and JSON \(p. 464\)](#)
- [Conventions Used in This Grammar \(p. 464\)](#)
- [Grammar \(p. 465\)](#)
- [Policy Grammar Notes \(p. 466\)](#)

The Policy Language and JSON

Policies are expressed in JSON. When a policy is submitted to IAM, it is first validated to make sure that the JSON syntax is correct. In this document, we do not provide a complete description of what constitutes valid JSON. However, here are some basic JSON rules:

- White space between individual entities is allowed.
- Values are enclosed in quotation marks. Quotation marks are optional for numeric and Boolean values.
- Many elements (for example, `action_string_list` and `resource_string_list`) can take a JSON array as a value. Arrays can take one or more values. If more than one value is included, the array is in square brackets ([and]) and comma-delimited, as in the following example:

`"Action" : ["ec2:Describe*", "ec2>List*"]`
- Basic JSON data types (Boolean, number, and string) are defined in [RFC 7159](#).

You can use a JSON validator to check the syntax of a policy. You can find a validator online, and many code editors and XML-editing tools include JSON validation features.

Conventions Used in This Grammar

The following conventions are used in this grammar:

- The following characters are JSON tokens and *are* included in policies:

`{ } [] " , :`
- The following characters are special characters in the grammar and are *not* included in policies:

`= < > () |`
- If an element allows multiple values, it is indicated using repeated values, a comma delimiter, and an ellipsis (...). Examples:

```
[<action_string>, <action_string>, ...]  
<principal_map> = { <principal_map_entry>, <principal_map_entry>, ... }
```

If multiple values are allowed, it is also valid to include only one value. For only one value, the trailing comma must be omitted. If the element takes an array (marked with [and]) but only one value is included, the brackets are optional. Examples:

```
"Action": [<action_string>]
```

"Action": <action_string>

- A question mark (?) following an element indicates that the element is optional. Example:

<version_block?>

However, be sure to refer to the notes that follow the grammar listing for details about optional elements.

- A vertical line (|) between elements indicates alternatives. In the grammar, parentheses define the scope of the alternatives. Example:

("Principal" | "NotPrincipal")

- Elements that must be literal strings are enclosed in double quotation marks (""). Example:

<version_block> = "Version" : ("2008-10-17" | "2012-10-17")

For additional notes, see [Policy Grammar Notes \(p. 466\)](#) following the grammar listing.

Grammar

The following listing describes the policy language grammar. For conventions used in the listing, see the preceding section. For additional information, see the notes that follow.

Note

This grammar describes policies marked with a version of 2008-10-17 and 2012-10-17. A Version policy element is different from a policy version. The Version policy element is used within a policy and defines the version of the policy language. A policy version, on the other hand, is created when you make changes to a customer managed policy in IAM. The changed policy doesn't overwrite the existing policy. Instead, IAM creates a new version of the managed policy. To learn more about the Version policy element see [IAM JSON Policy Elements: Version \(p. 426\)](#). To learn more about policy versions, see [the section called "Versioning IAM Policies" \(p. 335\)](#).

```
policy  = {
    <version_block?>
    <id_block?>
    <statement_block>
}

<version_block> = "Version" : ("2008-10-17" | "2012-10-17")

<id_block> = "Id" : <policy_id_string>

<statement_block> = "Statement" : [ <statement>, <statement>, ... ]

<statement> = {
    <sid_block?>,
    <principal_block?>,
    <effect_block>,
    <action_block>,
    <resource_block>,
    <condition_block?>
}

<sid_block> = "Sid" : <sid_string>

<effect_block> = "Effect" : ("Allow" | "Deny")

<principal_block> = ("Principal" | "NotPrincipal") : ("*" | <principal_map>)
```

```

<principal_map> = { <principal_map_entry>, <principal_map_entry>, ... }

<principal_map_entry> = ("AWS" | "Federated" | "Service") :
    [<principal_id_string>, <principal_id_string>, ...]

<action_block> = ("Action" | "NotAction") :
    ("*" | [<action_string>, <action_string>, ...])

<resource_block> = ("Resource" | "NotResource") :
    ("*" | [<resource_string>, <resource_string>, ...])

<condition_block> = "Condition" : { <condition_map> }
<condition_map> {
    <condition_type_string> : { <condition_key_string> : <condition_value_list> },
    <condition_type_string> : { <condition_key_string> : <condition_value_list> }, ...
}
<condition_value_list> = [<condition_value>, <condition_value>, ...]
<condition_value> = ("string" | "number" | "Boolean")

```

Policy Grammar Notes

- A single policy can contain an array of statements.
- Policies have a maximum size between 2048 characters and 10,240 characters, depending on what entity the policy is attached to. For more information, see [Limitations on IAM Entities and Objects \(p. 414\)](#). Policy size calculations do not include white space characters.
- Individual elements must not contain multiple instances of the same key. For example, you cannot include the `Effect` block twice in the same statement.
- Blocks can appear in any order. For example, `version_block` can follow `id_block` in a policy. Similarly, `effect_block`, `principal_block`, `action_block` can appear in any order within a statement.
- The `id_block` is optional in resource-based policies. It must *not* be included in identity-based policies.
- The `principal_block` element is required in resource-based policies (for example, in Amazon S3 bucket policies) and in trust policies for IAM roles. It must *not* be included in identity-based policies.
- Each string value (`policy_id_string`, `sid_string`, `principal_id_string`, `action_string`, `resource_string`, `condition_type_string`, `condition_key_string`, and the string version of `condition_value`) can have its own minimum and maximum length restrictions, specific allowed values, or required internal format.

Notes About String Values

This section provides additional information about string values that are used in different elements in a policy.

`action_string`

Consists of a service namespace, a colon, and the name of an action. Action names can include wildcards. Examples:

```

"Action": "ec2:StartInstances"

"Action": [
    "ec2:StartInstances",
    "ec2:StopInstances"
]

"Action": "cloudformation:/*"

"Action": "*"

```

```
"Action": [  
    "s3:Get*",  
    "s3>List*"  
]
```

policy_id_string

Provides a way to include information about the policy as a whole. Some services, such as Amazon SQS and Amazon SNS, use the `Id` element in reserved ways. Unless otherwise restricted by an individual service, `policy_id_string` can include spaces. Some services require this value to be unique within an AWS account.

Note

The `id_block` is allowed in resource-based policies, but not in identity-based policies.

There is no limit to the length, although this string contributes to the overall length of the policy, which is limited.

```
"Id": "Admin_Policy"  
"Id": "cd3ad3d9-2776-4ef1-a904-4c229d1642ee"
```

sid_string

Provides a way to include information about an individual statement. For IAM policies, basic alphanumeric characters (A-Z,a-z,0-9) are the only allowed characters in the `Sid` value. Other AWS services that support resource policies may have other requirements for the `Sid` value. For example, some services require this value to be unique within an AWS account, and some services allow additional characters such as spaces in the `Sid` value.

```
"Sid": "1"  
"Sid": "ThisStatementProvidesPermissionsForConsoleAccess"
```

principal_id_string

Provides a way to specify a principal using the [Amazon Resource Name \(ARN\)](#) (p. 410) of the AWS account, IAM user, IAM role, federated user, or assumed-role user. For an AWS account, you can also use the short form `AWS:accountnumber` instead of the full ARN. For all of the options including AWS services, assumed roles, and so on, see [Specifying a Principal](#) (p. 429).

Note that you can use `*` only to specify "everyone/anonymous." You cannot use it to specify part of a name or ARN.

resource_string

In most cases, consists of an [Amazon Resource Name \(ARN\)](#).

```
"Resource": "arn:aws:iam::123456789012:user/Bob"  
"Resource": "arn:aws:s3:::examplebucket/*"
```

condition_type_string

Identifies the type of condition being tested, such as `StringEquals`, `StringLike`, `NumericLessThan`, `DateGreaterThanOrEqual`, `Bool`, `BinaryEquals`, `IpAddress`, `ArnEquals`, etc. For a complete list of condition types, see [IAM JSON Policy Elements: Condition Operators](#) (p. 440).

```
"Condition": {
```

```
        "NumericLessThanEquals": {
            "s3:max-keys": "10"
        }
    }

    "Condition": {
        "Bool": {
            "aws:SecureTransport": "true"
        }
    }

    "Condition": {
        "StringEquals": {
            "s3:x-amz-server-side-encryption": "AES256"
        }
    }
}
```

condition_key_string

Identifies the condition key whose value will be tested to determine whether the condition is met. AWS defines a set of condition keys that are available in all AWS services, including `aws:principalType`, `aws:SecureTransport`, and `aws:userId`.

For a list of AWS condition keys, see [Available Global Condition Keys \(p. 477\)](#). For condition keys that are specific to a service, see the documentation for that service such as the following:

- [Specifying Conditions in a Policy](#) in the *Amazon Simple Storage Service Developer Guide*
- [IAM Policies for Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*.

```
"Condition": {
    "Bool": {
        "aws:SecureTransport": "true"
    }
}

"Condition": {
    "StringNotEquals": {
        "s3:x-amz-server-side-encryption": "AES256"
    }
}

"Condition": {
    "StringEquals": {
        "ec2:ResourceTag/purpose": "test"
    }
}
```

AWS Managed Policies for Job Functions

AWS managed policies for job functions are designed to closely align to common job functions in the IT industry. You can use these policies to easily grant the permissions needed to carry out the tasks expected of someone in a specific job function. These policies consolidate permissions for many services into a single policy that's easier to work with than having permissions scattered across many policies.

You can attach these policies for job functions to any group, user, or role.

Use Roles to Combine Services

Some of the policies use IAM service roles to help you take advantage of features found in other AWS services. These policies grant access to `iam:passrole`, which allows a user with the policy to pass a role

to an AWS service. This role delegates IAM permissions to the AWS service to carry out actions on your behalf.

You must create the roles according to your needs. For example, the Network Administrator policy allows a user with the policy to pass a role named "flow-logs-vpc" to the Amazon CloudWatch service. CloudWatch uses that role to log and capture IP traffic for VPCs created by the user.

To follow security best practices, the policies for job functions include filters that limit the names of valid roles that can be passed. This helps avoid granting unnecessary permissions. If your users do require the optional service roles, you must create a role that follows the naming convention specified in the policy. You then grant permissions to the role. Once that is done, the user can configure the service to use the role, granting it whatever permissions the role provides.

Keep Up to Date

These policies are all maintained by AWS and are kept up to date to include support for new services and new capabilities as they are added by AWS. These policies cannot be modified by customers. You can make a copy of the policy and then modify the copy, but that copy is not automatically updated as AWS introduces new services and API operations.

Job Functions

Names of policies

- [Administrator \(p. 469\)](#)
- [Billing \(p. 469\)](#)
- [Database Administrator \(p. 470\)](#)
- [Data Scientist \(p. 471\)](#)
- [Developer Power User \(p. 471\)](#)
- [Network Administrator \(p. 472\)](#)
- [System Administrator \(p. 472\)](#)
- [Security Auditor \(p. 473\)](#)
- [Support User \(p. 473\)](#)
- [View-Only User \(p. 473\)](#)

In the following sections, each policy's name is a link to the policy details page in the AWS Management Console. There you can see the policy document and review the permissions it grants.

[Administrator](#)

AWS managed policy name: [AdministratorAccess](#)

Use case: This user has full access and can delegate permissions to every service and resource in AWS.

Policy description: This policy grants all actions for all AWS services and for all resources in the account.

[Billing](#)

AWS managed policy name: [Billing](#)

Use case: This user needs to view billing information, set up payment, and authorize payment. The user can monitor the costs accumulated for each AWS service.

Policy description: This policy grants permissions for managing billing and costs. The permissions include viewing and modifying both budgets and payment methods.

Note

Before an IAM user can access the AWS Billing and Cost Management console with this policy, you must first enable Billing and Cost Management console access for the account. To do this, follow the instructions in [Step 1 of the tutorial about delegating access to the billing console \(p. 22\)](#).

Database Administrator

AWS managed policy name: [DatabaseAdministrator](#)

Use case: This user sets up, configures, and maintains databases in the AWS Cloud.

Policy description: This policy grants permissions to create, configure, and maintain databases. It includes access to all AWS database services, such as Amazon DynamoDB, Amazon ElastiCache, Amazon Relational Database Service (RDS), Amazon Redshift, and other supporting services.

This policy supports the ability to pass roles to AWS services. The policy grants `iam:GetRole` and `iam:PassRole` for only those roles named in the following table. For more information, see [Creating the Roles and Attaching the Policies \(Console\) \(p. 473\)](#) later in this topic.

Optional IAM service roles for the Database Administrator job function

Use case	Role name (* is a wildcard)	Service role type to select	Select this AWS managed policy
Allow the user to monitor RDS databases	rds-monitoring-role	Amazon RDS Role for Enhanced Monitoring	AmazonRDSEnhancedMonitoringRole
Allow AWS Lambda to monitor your database and access external databases	rdbms-lambda-access	Amazon EC2	AWSLambdaFullAccess
Allow Lambda to upload files to Amazon S3 and to Amazon Redshift clusters with DynamoDB	lambda_exec_role	AWS Lambda	Create a new managed policy as defined in the AWS Big Data Blog
Allow Lambda functions to act as triggers for your DynamoDB tables	lambda-dynamodb-*	AWS Lambda	AWSLambdaDynamoDBExecutionRole
Allow Lambda functions to access Amazon RDS in a VPC	lambda-vpc-execution-role	Create a role with a trust policy as defined in the AWS Lambda Developer Guide	AWSLambdaVPCAccessExecutionRole
Allow AWS Data Pipeline to access your AWS resources	DataPipelineDefaultRole	Create a role with a trust policy as defined in the AWS Data Pipeline Developer Guide	AWSDataPipelineRole
Allow your applications running on Amazon EC2 instances to access your AWS resources	DataPipelineDefaultRole	Create a role with a trust policy as defined in the AWS Data Pipeline Developer Guide	AmazonEC2RoleforDataPipelineRole

Data Scientist

AWS managed policy name: [DataScientist](#)

Use case: This user runs Hadoop jobs and queries. The user also accesses and analyzes information for data analytics and business intelligence.

Policy description: This policy grants permissions to create, manage, and run queries on an Amazon EMR cluster and perform data analytics with tools such as Amazon QuickSight. The policy includes access to AWS Data Pipeline, Amazon EC2, Amazon Elasticsearch Service, Amazon Elastic File System, Amazon EMR, Amazon Kinesis, Amazon Kinesis Data Analytics, Amazon Machine Learning, Amazon RDS, and Amazon Redshift.

This job function supports the ability to pass roles to AWS services. The policy grants `iam:GetRole` and `iam:PassRole` for only those roles named in the following table. For more information, see [Creating the Roles and Attaching the Policies \(Console\) \(p. 473\)](#) later in this topic.

Optional IAM service roles for the Data Scientist job function

Use case	Role name (* is a wildcard)	Service role type to select	AWS managed policy to select
Allow Amazon EC2 instances access to services and resources suitable for clusters	EMR-EC2_DefaultRole	Amazon EMR for EC2	AmazonElasticMapReduceforEC2Role
Allow Amazon EMR access to access the Amazon EC2 service and resources for clusters	EMR_DefaultRole	Amazon EMR	AmazonElasticMapReduceRole
Allow Kinesis Kinesis Data Analytics to access streaming data sources	kinesis-*	Create a role with a trust policy as defined in the AWS Big Data Blog .	See the AWS Big Data Blog , which outlines four possible options depending on your use case
Allow AWS Data Pipeline to access your AWS resources	DataPipelineDefaultRole	Create a role with a trust policy as defined in the AWS Data Pipeline Developer Guide	AWSDataPipelineRole
Allow your applications running on Amazon EC2 instances to access your AWS resources	DataPipelineDefaultRole	Create a role with a trust policy as defined in the AWS Data Pipeline Developer Guide	AmazonEC2RoleforDataPipelineRole

Developer Power User

AWS managed policy name: [PowerUserAccess](#)

Use case: This user performs application development tasks and can create and configure resources and services that support AWS aware application development.

Policy description: This policy grants permissions to view, read, and write permissions for a variety of AWS services intended for application development, including Amazon API Gateway, Amazon AppStream, Amazon CloudSearch, AWS CodeCommit, AWS CodeDeploy, AWS CodePipeline, AWS Device Farm, Amazon DynamoDB, Amazon Elastic Compute Cloud, Amazon Elastic Container Service (ECS), AWS

Lambda, Amazon RDS, Amazon Route 53, Amazon Simple Storage Service (S3), Amazon Simple Email Service (SES), Amazon Simple Queue Service (SQS), and Amazon Simple Workflow Service (SWF).

Network Administrator

AWS managed policy name: [NetworkAdministrator](#)

Use case: This user is tasked with setting up and maintaining AWS network resources.

Policy description: This policy grants permissions to create and maintain network resources in Amazon EC2, Route 53, Amazon Virtual Private Cloud (VPC), and AWS Direct Connect.

This job function requires the ability to pass roles to AWS services. The policy grants `iam:GetRole` and `iam:PassRole` for only those roles named in the following table. For more information, see [Creating the Roles and Attaching the Policies \(Console\) \(p. 473\)](#) later in this topic.

Optional IAM service roles for the Network Administrator job function

Use case	Role name (* is a wildcard)	Service role type to select	AWS managed policy to select
Allows Amazon VPC to create and manage logs in CloudWatch Logs on the user's behalf to monitor IP traffic going in and out of your VPC	flow-logs-*	Create a role with a trust policy as defined in the Amazon VPC User Guide	This use case does not have an existing AWS managed policy, but the documentation lists the required permissions. See Amazon VPC User Guide .

System Administrator

AWS managed policy name: [SystemAdministrator](#)

Use case: This user sets up and maintains resources for development operations.

Policy description: This policy grants permissions to create and maintain resources across a large variety of AWS services, including AWS CloudTrail, Amazon CloudWatch, AWS CodeCommit, AWS CodeDeploy, AWS Config, AWS Directory Service, Amazon EC2, AWS Identity and Access Management, AWS Key Management Service, AWS Lambda, Amazon RDS, Route 53, Amazon S3, Amazon SES, Amazon SQS, AWS Trusted Advisor, and Amazon VPC.

This job function requires the ability to pass roles to AWS services. The policy grants `iam:GetRole` and `iam:PassRole` for only those roles named in the following table. For more information, see [Creating the Roles and Attaching the Policies \(Console\) \(p. 473\)](#) later in this topic.

Optional IAM service roles for the System Administrator job function

Use case	Role name (* is a wildcard)	Service role type to select	AWS managed policy to select
Allow apps running in EC2 instances in an Amazon ECS cluster to access Amazon ECS	ecr-sysadmin-*	Amazon EC2 Role for EC2 Container Service	AmazonEC2ContainerServiceforEC2
Allow a user to monitor databases	rds-monitoring-role	Amazon RDS Role for Enhanced Monitoring	AmazonRDSEnhancedMonitoringRo

Use case	Role name (* is a wildcard)	Service role type to select	AWS managed policy to select
Allow apps running in EC2 instances to access AWS resources.	ec2-sysadmin-*	Amazon EC2	Sample policy for role that grants access to an S3 bucket as shown in the Amazon EC2 User Guide for Linux Instances ; customize as needed
Allow Lambda to read DynamoDB streams and write to CloudWatch Logs	lambda-sysadmin-*	AWS Lambda	AWSLambdaDynamoDBExecutionRole

Security Auditor

AWS managed policy name: [SecurityAudit](#)

Use case: This user monitors accounts for compliance with security requirements. This user can access logs and events to investigate potential security breaches or potential malicious activity.

Policy description: This policy grants permissions to view configuration data for many AWS services and to review their logs.

Support User

AWS managed policy name: [SupportUser](#)

Use case: This user contacts AWS Support, creates support cases, and views the status of existing cases.

Policy description: This policy grants permissions to create and update AWS Support cases.

View-Only User

AWS managed policy name: [ViewOnlyAccess](#)

Use case: This user can view a list of AWS resources and basic metadata in the account across all services. The user cannot read resource content or metadata that goes beyond the quota and list information for resources.

Policy description: This policy grants `List*` and `Describe*` access to resources for every AWS service.

Creating the Roles and Attaching the Policies (Console)

Several of the previously listed policies grant the ability to configure AWS services with roles that enable those services to perform operations on your behalf. The job function policies either specify exact role names that you must use or at least include a prefix that specifies the first part of the name that can be used. To create one of these roles, perform the steps in the following procedure.

To create a role for an AWS service (IAM console)

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 of the IAM console, choose **Roles**, and then choose **Create role**.
3. Choose the **AWS Service** role type, and then choose the service that you want to allow to assume this role.

4. Choose the use case for your service. If the specified service has only one use case, it is selected for you. Use cases are defined by the service to include the trust policy that the service requires. Then choose **Next: Permissions**.
5. Choose one or more permissions policies to attach to the role. Depending on the use case that you selected, the service might do any of the following:
 - Define the permissions that the role uses
 - Allow you to choose from a limited set of permissions
 - Allow you to choose from any permissions
 - Allow you to select no policies at this time, create the policies later, and then attach them to the role

Select the box next to the policy that assigns the permissions that you want the users to have, and then choose **Next: Review**.

Note

The permissions that you specify are available to any entity that uses the role. By default, a role has no permissions.

6. For **Role name**, the degree of role name customization is defined by the service. If the service defines the role's name, this option is not editable. In other cases, the service might define a prefix for the role and allow you to type an optional suffix. Some services allow you to specify the entire name of your role.

If possible, type a role name or role name suffix to help you identify the purpose of this role. Role names must be unique within your AWS account. They are not distinguished by case. For example, you cannot create roles named both **PRODROLE** and **prodrole**. Because various entities might reference the role, you cannot edit the name of the role after it has been created.

7. (Optional) For **Role description**, type a description for the new role.
8. Review the role and then choose **Create role**.

Example 1: Configuring a User as a Database Administrator (Console)

This example shows the steps required to configure Alice, an IAM user, as a [Database Administrator \(p. 470\)](#). You use the information in first row of the table in that section and allow the user to enable Amazon RDS monitoring. You attach the [DatabaseAdministrator](#) policy to Alice's IAM user so that she can manage the Amazon database services. That policy also enables Alice to pass a role called `rds-monitoring-role` to the Amazon RDS service that allows the service to monitor the RDS databases on her behalf.

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Choose **Policies** and then type **database** in the search box.
3. Select the check box for the **DatabaseAdministrator** policy, choose **Policy actions**, and then choose **Attach**.
4. In the list of users, select **Alice** and then choose **Attach policy**. Alice now can administer AWS databases. However, to allow Alice to monitor those databases, you must configure the service role.
5. In the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.
6. Choose the **AWS Service** role type, and then choose **Amazon RDS**.
7. Choose the **Amazon RDS Role for Enhanced Monitoring** use case.
8. Amazon RDS defines the permissions for your role. Choose **Next: Review** to continue.

9. The role name must be one of those specified by the DatabaseAdministrator policy that Alice now has. One of those is **rds-monitoring-role**. Type that for the **Role name**.
10. (Optional) For **Role description**, type a description for the new role.
11. After you review the details, choose **Create role**.
12. Alice can now enable **RDS Enhanced Monitoring** in the **Monitoring** section of the Amazon RDS console. For example, she might do this when she creates a DB instance, creates a read replica, or modifies a DB instance. She must type the role name she created (rds-monitoring-role) in the **Monitoring Role** box when she sets **Enable Enhanced Monitoring** to **Yes**.

Example 2: Configuring a User as a Network Administrator (Console)

This example shows the steps required to configure Juan, an IAM user, as a [Network Administrator \(p. 472\)](#). It uses the information in the table in that section to allow Juan to monitor IP traffic going to and from a VPC. It also allows Juan to capture that information in the logs in CloudWatch Logs. You attach the [NetworkAdministrator](#) policy to Juan's IAM user so that he can configure AWS network resources. That policy also enables Juan to pass a role whose name begins with `flow-logs*` to Amazon EC2 when you create a flow log. In this scenario, unlike Example 1, there isn't a predefined service role type, so you must perform a few steps differently.

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, choose **Policies** and then type **network** in the search box.
3. Select the check box next to **NetworkAdministrator** policy, choose **Policy actions**, and then choose **Attach**.
4. In the list of users, select the check box next to **Juan** and then choose **Attach policy**. Juan now can administer AWS network resources. However, to enable monitoring of IP traffic in your VPC, you must configure the service role.
5. Because the service role you need to create doesn't have a predefined managed policy, you must first create it. In the navigation pane, choose **Policies**, then choose **Create policy**.
6. Choose the **JSON** tab and copy the text from the following JSON policy document. Paste this text into the **JSON** text box.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "logs:CreateLogGroup",  
                "logs:CreateLogStream",  
                "logs:PutLogEvents",  
                "logs:DescribeLogGroups",  
                "logs:DescribeLogStreams"  
            ],  
            "Effect": "Allow",  
            "Resource": "*"  
        }  
    ]  
}
```

7. When you are finished, choose **Review policy**. The [Policy Validator \(p. 321\)](#) reports any syntax errors.

Note

You can switch between the **Visual editor** and **JSON** tabs any time. However, if you make changes or choose **Review policy** in the **Visual editor** tab, IAM might restructure your policy to optimize it for the visual editor. For more information, see [Policy Restructuring \(p. 387\)](#).

8. On the **Review** page, type **vpc-flow-logs-policy-for-service-role** for the policy name. Review the policy **Summary** to see the permissions granted by your policy, and then choose **Create policy** to save your work.

The new policy appears in the list of managed policies and is ready to attach.
9. In the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.
10. Choose the **AWS Service** role type, and then choose **Amazon EC2**.
11. Choose the **Amazon EC2** use case.
12. On the **Attach permissions policies** page, choose the policy you created earlier, **vpc-flow-logs-policy-for-service-role**, and then choose **Next: Review**.
13. The role name must be permitted by the NetworkAdministrator policy that Juan now has. Any name that begins with **flow-logs-** is allowed. For this example, type **flow-logs-for-juan** for the **Role name**.
14. (Optional) For **Role description**, type a description for the new role.
15. After you review the details, choose **Create role**.
16. Now you can configure the trust policy required for this scenario. On the **Roles** page, choose the **flow-logs-for-juan** role (the name, not the check box). On the details page for your new role, choose the **Trust relationships** tab, and then choose **Edit trust relationship**.
17. Change the "Service" line to read as follows, replacing the entry for `ec2.amazonaws.com`:

```
"Service": "vpc-flow-logs.amazonaws.com"
```

18. Juan can now create flow logs for a VPC or subnet in the Amazon EC2 console. When you create the flow log, specify the **flow-logs-for-juan** role. That role has the permissions to create the log and write data to it.

AWS Global and IAM Condition Context Keys

The **Condition** element of an JSON policy in IAM allows you to test the value of keys that are included in the evaluation context of all AWS API requests. These keys provide information about the request itself, or the resources that the request references. You can check that keys have specified values before allowing the action requested by the user. This gives you granular control over when your JSON policy statements match or don't match an incoming API request. For information about how to use the **Condition** element in an JSON policy, see [IAM JSON Policy Elements: Condition \(p. 438\)](#).

This topic describes the globally available keys (with an "aws:" prefix), as well as the keys defined and provided by the IAM service (with an "iam:" prefix). Several other AWS services also provide service-specific keys that are relevant to the actions and resources defined by that service. For more information, see [AWS Service Actions and Condition Context Keys for Use in IAM Policies \(p. 485\)](#). The documentation for a service that supports condition keys often has additional information. For example, for information about keys that you can use in policies for Amazon S3 resources, see [Amazon S3 Policy Keys](#) in the [Amazon Simple Storage Service Developer Guide](#).

Note

If you use condition keys that are available only in some scenarios (such as `aws:SourceIp` and `aws:SourceVpc`) you can use the [IfExists \(p. 446\)](#) versions of the comparison operators. If the condition keys are missing from a request context (and you haven't set `IfExists`), the policy engine can fail the evaluation. For example, if you want to write a policy that restricts access from a particular IP range or from a particular VPC, you can construct the conditions as follows:

```
"Condition": {"IpAddressIfExists": {"aws:SourceIp": ["xxx"]}},  
"StringEqualsIfExists": {"aws:SourceVpc": ["yyy"]}}
```

This condition matches (1) if the `aws:SourceIp` context key exists and has the value `xxx` or (2) if the `aws:SourceVpc` context key exists and has the value `yyy`. If either or both keys do not

exist, the condition still matches. The test is only made if the specified key exists in the request context. If it is not there, it is treated as "I don't care".

Topics

- [Available Global Condition Keys \(p. 477\)](#)
- [Available Keys for IAM \(p. 480\)](#)
- [Available Keys for Web Identity Federation \(p. 480\)](#)
- [Available Keys for SAML-Based Federation \(p. 481\)](#)

Available Global Condition Keys

AWS provides the following predefined condition keys for all AWS services that support IAM for access control:

aws:currentTime

Works with [date operators \(p. 443\)](#).

The current date and time to check for date/time conditions.

aws:epochTime

Works with [date operators \(p. 443\)](#).

The current date and time in epoch or UNIX time to check for date/time conditions.

aws:tokenIssueTime

Works with [date operators \(p. 443\)](#).

To check the date/time that temporary security credentials were issued. This key is only present in requests that are signed using temporary security credentials. For more information about temporary security credentials, see [Temporary Security Credentials \(p. 231\)](#).

aws:multiFactorAuthPresent

Works with [boolean operators \(p. 443\)](#).

To check whether multi-factor authentication (MFA) was used to validate the temporary security credentials that made the current request. This key is present in the request context only when the user uses temporary credentials to call the API. Such credentials are used with IAM roles, federated users, IAM users with credentials from `sts:GetSessionToken`, and users of the AWS Management Console. (The console uses temporary credentials that are generated on the users' behalf in the background.) The `aws:MultiFactorAuthPresent` key is never present when an API or CLI command is called with long-term credentials, such as standard access key pairs. Therefore we recommend that when you check for this key that you use the [...IfExists \(p. 446\)](#) versions of the condition operators.

It is important to understand that the following Condition element is **not** a reliable way to check for the use of MFA:

```
##### THIS EXAMPLE DOES NOT WORK #####
"Condition" : { "Bool" : { "aws:MultiFactorAuthPresent" : false } }
```

This is because when long term credentials are used, the `aws:MultiFactorAuthPresent` key is not present in the request and the test *always* fails, resulting in a non-match. Instead, we recommend that you use the [BoolIfExists \(p. 446\)](#) operator to check the value. For example::

```
"Condition" : { "BoolIfExists" : { "aws:MultiFactorAuthPresent" : false } }
```

This operator indicates that the statement matches either if the value exists and has the value of "false" **or** if the value does not exist. ...**IfExists** says that in effect you don't care whether the key to check does not exist in the context of the request; its absence will not result in a nonmatch.

Do not use a policy construct similar to the following to check whether the MFA key is present:

```
##### THIS EXAMPLE DOES NOT WORK #####
"Action" : "Deny",
"Condition" : { "Null" : { "aws:MultiFactorAuthPresent" : true } }
```

You might expect the previous example to deny access if MFA is not used. However, when you make an API request with long-term credentials (access keys), the MFA condition context keys are *always* missing. Consequently, testing for MFA this way always results in denied access to long-term credentials.

aws:MultiFactorAuthAge

Works with [numeric operators \(p. 442\)](#).

To check how long ago (in seconds) the MFA-validated security credentials making the request were issued using multi-factor authentication (MFA). If MFA was not used, this key is not present. See [Using Multi-Factor Authentication \(MFA\) in AWS \(p. 91\)](#). Therefore, this is another key with which you should consider using the ...[IfExists \(p. 446\)](#) versions of the comparison operators. This ensures that the results of the comparison are what you expect even when the key is not present in the request context.

aws:PrincipalType

Works with [string operators \(p. 441\)](#).

To check the type of principal (user, account, federated user, etc.) for the current request.

aws:Referer

Works with [string operators \(p. 441\)](#).

To check who referred the client browser to the address the request is being sent to. It is only supported by some services, such as [Amazon S3, as a service that can be directly addressed by a web browser](#). The value comes from the referer header in the HTTPS request made to AWS.

Warning

This key should be used carefully: `aws:referer` allows Amazon S3 bucket owners to help prevent their content from being served up by unauthorized third-party sites to standard web browsers (for more information, see the link above). Since the `aws:referer` value is provided by the caller in an http header, unauthorized parties can use modified or custom browsers to provide any `aws:referer` value that they choose. As a result, `aws:referer` should not be used to prevent unauthorized parties from making direct AWS requests. It is offered only to allow customers to protect their digital content, stored in Amazon S3, from being referenced on unauthorized, third-party sites.

aws:RequestTag/tag-key

Works with [string operators \(p. 441\)](#).

This context key is formatted "`aws:RequestTag/tag-key":"tag-value`" where `tag-key` and `tag-value` are a tag key and value pair.

To check a tag and its value in an AWS request. For example, you could check to see that the request includes the tag key "Dept" and that it has the value "Accounting".

This AWS condition key was [introduced for Amazon EC2](#) and is supported by a limited number of additional services. Check your service to see whether it supports using this condition key.

aws:SecureTransport

Works with [boolean operators \(p. 443\)](#).

To check whether the request was sent using SSL.

aws:SourceAccount

Works with [string operators \(p. 441\)](#).

To check that the source of the request comes from a specific account. For example, if an S3 bucket in your account is configured to deliver object creation events to an SNS topic, use this condition key to check that S3 is not being used as a confused deputy. S3 tells SNS the account that the bucket belongs to.

This condition key is available for only some services.

aws:SourceArn

Works with [ARN operators \(p. 445\)](#).

To check the source of the request, using the [Amazon Resource Name \(ARN\) \(p. 410\)](#) of the source.

This condition key is available for only some services.

aws:SourceIp

Works with [IP address operators \(p. 444\)](#).

To check the requester's IP address, see [IP Address Condition Operators \(p. 444\)](#).

Note

The `aws:SourceIp` condition key should be used in a JSON policy only for IAM users, groups, roles, or federated users that make API calls from within the specified IP range. This policy denies access to an AWS service that makes calls on your behalf. For example, if you have a [service role \(p. 135\)](#) that allows AWS CloudFormation to call Amazon EC2 to stop an instance, then the request is denied because the target service (EC2) sees the IP address of the calling service (CloudFormation) rather than the IP address of the originating user. There is no way to pass the originating IP address through a calling service to the target service for evaluation in an JSON policy.

If the request comes from a host that uses an Amazon VPC endpoint, then the `aws:SourceIp` key is not available. You should instead use a VPC-specific key. For more information, see [VPC Endpoints - Controlling the Use of Endpoints](#) in the [Amazon VPC User Guide](#).

aws:SourceVpc

Works with [string operators \(p. 441\)](#).

To restrict access to a specific VPC. For more information, see [Restricting Access to a Specific VPC](#) in the [Amazon Simple Storage Service Developer Guide](#). (This condition key is supported for traffic to an AWS service over a VPC endpoint.)

aws:SourceVpce

Works with [string operators \(p. 441\)](#).

To restrict access to a specific VPC endpoint. For more information, see [Restricting Access to a Specific VPC Endpoint](#) in the [Amazon Simple Storage Service Developer Guide](#).

aws:TagKeys

Works with [string operators \(p. 441\)](#).

This context key is formatted "aws:TagKeys":"tag-key" where *tag-key* is a list of tag keys without values (for example, ["Dept", "Cost-Center"]).

To check the tag keys that are present in an AWS request. This AWS condition key was [introduced for Amazon EC2](#) and is supported by a limited number of additional services. Check your service to see whether it supports using this condition key.

aws:UserAgent

Works with [string operators \(p. 441\)](#).

To check the requester's client application.

aws:userid

Works with [string operators \(p. 441\)](#).

To check the requester's user ID.

aws:username

Works with [string operators \(p. 441\)](#).

To check the requester's user name.

Available Keys for IAM

You can use the following condition keys in policies that control access to IAM resources:

iam:PolicyArn

Works with [ARN operators \(p. 445\)](#).

To check the Amazon Resource Name (ARN) of a managed policy in requests that involve a managed policy. For more information, see [Controlling Access to Policies \(p. 291\)](#).

Available Keys for Web Identity Federation

If you are using web identity federation to give temporary security credentials to users who have been authenticated using an identity provider (IdP) such as Login with Amazon, Amazon Cognito, Google, or Facebook, additional condition keys are available when the temporary security credentials are used to make a request. These keys let you write policies that make sure that federated users can get access only to resources that are associated with a specific provider, app, or user.

aws:FederatedProvider

Works with [string operators \(p. 441\)](#).

The FederatedProvider key identifies which of the IdPs was used to authenticate the user. For example, if the user authenticated using Amazon Cognito, the key would contain cognito-identity.amazonaws.com. Similarly, if the user authenticated using Login with Amazon, the key would contain the value www.amazon.com. You might use the key in a resource policy like the following, which uses the aws:FederatedProvider key as a policy variable in the ARN of a resource. The policy allows any user who has been authenticated using an IdP to get objects out of a folder in an Amazon S3 bucket that's specific to the provider they used to authenticate with.

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

```
        "Effect": "Allow",
        "Principal": "*",
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::BUCKET-NAME/${aws:FederatedProvider}/*"
    }
}
```

Application ID and User ID

Works with [string operators \(p. 441\)](#).

You can also use two keys that provide a unique identifier for the user and an identifier for the application or site that the user authenticated with. These keys have the following IdP-specific names:

- For Amazon Cognito users, the keys are `cognito-identity.amazonaws.com:aud` (for the identity pool ID) and `cognito-identity.amazonaws.com:sub` (for the user ID).
- For Login With Amazon users, the keys are `www.amazon.com:app_id` and `www.amazon.com:user_id`
- For Facebook users, the keys are `graph.facebook.com:app_id` and `graph.facebook.com:id`
- For Google users, the keys are `accounts.google.com:aud` (for the app ID) and `accounts.google.com:sub` (for the user ID).

The amr Key in Amazon Cognito

Works with [string operators \(p. 441\)](#).

If you are using Amazon Cognito for web identity federation, the `cognito-identity.amazonaws.com:amr` key (Authenticated Methods Reference) in a trust policy includes login information about the user. The key is multivalued, meaning that you test it in a policy using [condition set operators \(p. 447\)](#). The key can contain the following values:

- If the user is unauthenticated, the key contains only `unauthenticated`.
- If the user is authenticated, the key contains the value `authenticated` and the name of the login provider used in the call (`graph.facebook.com`, `accounts.google.com`, or `www.amazon.com`).

As an example, the following condition in the trust policy for an Amazon Cognito role tests whether the user is unauthenticated:

```
"Condition": {
    "StringEquals":
        { "cognito-identity.amazonaws.com:aud": "us-east-2:identity-pool-id" },
    "ForAnyValue:StringLike":
        { "cognito-identity.amazonaws.com:amr": "unauthenticated" }
}
```

More Information About Web Identity Federation

For more information about web identity federation, see the following:

- [Amazon Cognito Overview](#) in the *AWS Mobile SDK for Android Developer Guide* guide
- [Amazon Cognito Overview](#) in the *AWS Mobile SDK for iOS Developer Guide* guide
- [About Web Identity Federation \(p. 143\)](#)

Available Keys for SAML-Based Federation

If you are working with [SAML-based federation](#), you can include additional condition keys in the policy.

Trust Policies

In the trust policy of a role, you can include the following keys, which help you establish whether the caller is allowed to assume the role. Except for `saml:doc`, all the values are derived from the SAML assertion. Items in the list that are marked with an asterisk (*) are available in the console UI to create conditions. Items marked with [] can have a value that is a list of the specified type.

`saml:aud`

Works with [string operators \(p. 441\)](#).

An endpoint URL to which SAML assertions are presented. The value for this key comes from the SAML Recipient field in the assertion, **not** the Audience field.

`saml:cn[]`

Works with [string operators \(p. 441\)](#).

This is an eduOrg attribute.

`saml:doc*`

Works with [string operators \(p. 441\)](#).

This represents the principal that was used to assume the role. The format is `account-ID/provider-friendly-name`, such as `123456789012/SAMLProviderName`. The `account-ID` value refers to the account that owns the [SAML provider \(p. 158\)](#).

`saml:edupersonaffiliation[]`

Works with [string operators \(p. 441\)](#).

This is an eduPerson attribute.

`saml:edupersonassurance[]`

Works with [string operators \(p. 441\)](#).

This is an eduPerson attribute.

`saml:edupersonentitlement[]*`

Works with [string operators \(p. 441\)](#).

This is an eduPerson attribute.

`saml:edupersonnickname[]`

Works with [string operators \(p. 441\)](#).

This is an eduPerson attribute.

`saml:edupersonorgdn*`

Works with [string operators \(p. 441\)](#).

This is an eduPerson attribute.

`saml:edupersonorgunitdn[]`

Works with [string operators \(p. 441\)](#).

This is an eduPerson attribute.

`saml:edupersonprimaryaffiliation`

Works with [string operators \(p. 441\)](#).

This is an eduPerson attribute.

saml:edupersonprimaryorgunitdn

Works with [string operators \(p. 441\)](#).

This is an eduPerson attribute.

saml:edupersonprincipalname

Works with [string operators \(p. 441\)](#).

This is an eduPerson attribute.

saml:edupersonscopedaffiliation[]

Works with [string operators \(p. 441\)](#).

This is an eduPerson attribute.

saml:edupersontargetedid[]

Works with [string operators \(p. 441\)](#).

This is an eduPerson attribute.

saml:eduorghomepageuri[]

Works with [string operators \(p. 441\)](#).

This is an eduOrg attribute.

saml:eduorgidentityauthnpolicyuri[]

Works with [string operators \(p. 441\)](#).

This is an eduOrg attribute.

saml:eduorglegalname[]

Works with [string operators \(p. 441\)](#).

This is an eduOrg attribute.

saml:eduorgsuperioruri[]

Works with [string operators \(p. 441\)](#).

This is an eduOrg attribute.

saml:eduorgwhitepagesuri[]

Works with [string operators \(p. 441\)](#).

This is an eduOrg attribute.

saml:namequalifier*

Works with [string operators \(p. 441\)](#).

A hash value based on the concatenation of the Issuer response value (`saml:iss`), the AWS account ID and the friendly name (the last part of the ARN) of the SAML provider in IAM, separated by a '/' character. The concatenation of the account ID and friendly name of the SAML provider is available to IAM policies as the key `saml:doc`. For more information, see [Uniquely Identifying Users in SAML-Based Federation \(p. 152\)](#).

saml:iss*

Works with [string operators \(p. 441\)](#).

The issuer, which is represented by a URN.

saml:sub*

Works with [string operators \(p. 441\)](#).

This is the subject of the claim, which includes a value that uniquely identifies an individual user within an organization (for example, `_cbb88bf52c2510eabe00c1642d4643f41430fe25e3`).

saml:sub_type*

Works with [string operators \(p. 441\)](#).

This key can have the value "persistent", "transient", or consist of the full Format URI from the Subject and NameID elements used in your SAML assertion. A value of "persistent" indicates that the value in saml:sub is the same for a user between sessions. If the value is "transient", the user has a different saml:sub value for each session. For information about the NameID element's Format attribute, see [Configuring SAML Assertions for the Authentication Response \(p. 163\)](#).

For general information about eduPerson and eduOrg attributes, see the [Internet2 website](#). For a list of eduPerson attributes, see [eduPerson Object Class Specification \(201203\)](#).

Condition keys whose type is a list can include multiple values. To create conditions in the policy for list values, you can use [set operators \(p. 447\)](#) (ForAllValues, ForAnyValue). For example, to allow any user whose affiliation is "faculty", "staff", or "employee" (but not "student", "alum", or other possible affiliations), you might use a condition like the following:

```
"Condition": {  
    "ForAllValues:StringLike": {  
        "saml:edupersonaffiliation": [ "faculty", "staff", "employee" ]  
    }  
}
```

Permission (Access) Policies

In the permission policy of a role for SAML federation that defines what users are allowed to access in AWS, you can include the following keys:

saml:namequalifier

Works with [string operators \(p. 441\)](#).

This contains a hash value that represents the combination of the saml:doc and saml:iss values. It is used as a namespace qualifier; the combination of saml:namequalifier and saml:sub uniquely identifies a user.

saml:sub

Works with [string operators \(p. 441\)](#).

This is the subject of the claim, which includes a value that uniquely identifies an individual user within an organization (for example, `_cbb88bf52c2510eabe00c1642d4643f41430fe25e3`).

saml:sub_type

Works with [string operators \(p. 441\)](#).

This key can have the value "persistent", "transient", or consist of the full Format URI from the Subject and NameID elements used in your SAML assertion. A value of "persistent" indicates that the value in saml:sub is the same for a user between sessions. If the value is "transient", the user has a different saml:sub value for each session. For information about the NameID element's Format attribute, see [Configuring SAML Assertions for the Authentication Response \(p. 163\)](#).

For more information about using these keys, see [About SAML 2.0-based Federation \(p. 149\)](#).

AWS Service Actions and Condition Context Keys for Use in IAM Policies

Each AWS service can provide actions and condition context keys for use in IAM policies. Not all API actions defined by a service can be used in an IAM policy, and a service might define some permissions that don't directly correspond to an API action. Use this list to determine which actions can be used as permissions in an IAM policy.

- Use actions found in these lists in the `Action` element of an IAM policy to allow or deny what a user can do within a service. For more information about the `Action` element, see [Action in the IAM Policy Element Reference](#).
- Use the context keys in these lists in the `Condition` element of an IAM policy to allow or deny access only when specified values are present. For more information about the `Condition` element, see [IAM JSON Policy Elements: Condition \(p. 438\)](#).
- For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#).

AWS services with actions and/or condition context keys:

- [Actions and Condition Context Keys for Alexa for Business \(p. 488\)](#)
- [Actions and Condition Context Keys for Amazon API Gateway \(p. 489\)](#)
- [Actions and Condition Context Keys for Application Auto Scaling \(p. 489\)](#)
- [Actions and Condition Context Keys for Application Discovery \(p. 490\)](#)
- [Actions and Condition Context Keys for Amazon AppStream \(p. 490\)](#)
- [Actions and Condition Context Keys for AWS Artifact \(p. 491\)](#)
- [Actions and Condition Context Keys for Amazon Athena \(p. 492\)](#)
- [Actions and Condition Context Keys for Auto Scaling \(p. 493\)](#)
- [Actions and Condition Context Keys for Auto Scaling Plans \(p. 494\)](#)
- [Actions and Condition Context Keys for AWS Batch \(p. 495\)](#)
- [Actions and Condition Context Keys for AWS Billing \(p. 495\)](#)
- [Actions and Condition Context Keys for AWS Budget Service \(p. 496\)](#)
- [Actions and Condition Context Keys for AWS Certificate Manager \(p. 496\)](#)
- [Actions and Condition Context Keys for Amazon Chime \(p. 497\)](#)
- [Actions and Condition Context Keys for Amazon AWS Cloud Contact Center \(p. 498\)](#)
- [Actions and Condition Context Keys for Amazon Cloud Directory \(p. 498\)](#)
- [Actions and Condition Context Keys for AWS Cloud9 \(p. 500\)](#)
- [Actions and Condition Context Keys for AWS CloudFormation \(p. 501\)](#)
- [Actions and Condition Context Keys for Amazon CloudFront \(p. 502\)](#)
- [Actions and Condition Context Keys for AWS CloudHSM \(p. 503\)](#)
- [Actions and Condition Context Keys for Amazon CloudSearch \(p. 503\)](#)
- [Actions and Condition Context Keys for AWS CloudTrail \(p. 504\)](#)
- [Actions and Condition Context Keys for Amazon CloudWatch \(p. 505\)](#)
- [Actions and Condition Context Keys for Amazon CloudWatch Events \(p. 506\)](#)
- [Actions and Condition Context Keys for Amazon CloudWatch Logs \(p. 506\)](#)
- [Actions and Condition Context Keys for AWS Code Signing for Amazon FreeRTOS \(p. 507\)](#)
- [Actions and Condition Context Keys for AWS CodeBuild \(p. 508\)](#)

- [Actions and Condition Context Keys for AWS CodeCommit \(p. 508\)](#)
- [Actions and Condition Context Keys for AWS CodeDeploy \(p. 510\)](#)
- [Actions and Condition Context Keys for AWS CodePipeline \(p. 511\)](#)
- [Actions and Condition Context Keys for AWS CodeStar \(p. 512\)](#)
- [Actions and Condition Context Keys for Amazon Cognito Identity \(p. 512\)](#)
- [Actions and Condition Context Keys for Amazon Cognito Sync \(p. 513\)](#)
- [Actions and Condition Context Keys for Amazon Cognito User Pools \(p. 514\)](#)
- [Actions and Condition Context Keys for Amazon Comprehend \(p. 516\)](#)
- [Actions and Condition Context Keys for AWS Config \(p. 516\)](#)
- [Actions and Condition Context Keys for AWS Cost and Usage Report \(p. 517\)](#)
- [Actions and Condition Context Keys for AWS Cost Explorer Service \(p. 517\)](#)
- [Actions and Condition Context Keys for Data Pipeline \(p. 518\)](#)
- [Actions and Condition Context Keys for AWS Database Migration Service \(p. 519\)](#)
- [Actions and Condition Context Keys for AWS Device Farm \(p. 520\)](#)
- [Actions and Condition Context Keys for AWS Direct Connect \(p. 521\)](#)
- [Actions and Condition Context Keys for AWS Directory Service \(p. 522\)](#)
- [Actions and Condition Context Keys for Amazon DynamoDB \(p. 523\)](#)
- [Actions and Condition Context Keys for Amazon DynamoDB Accelerator \(DAX\) \(p. 524\)](#)
- [Actions and Condition Context Keys for Amazon EC2 \(p. 525\)](#)
- [Actions and Condition Context Keys for Amazon EC2 Container Registry \(p. 532\)](#)
- [Actions and Condition Context Keys for Amazon EC2 Container Service \(p. 533\)](#)
- [Actions and Condition Context Keys for AWS Elastic Beanstalk \(p. 534\)](#)
- [Actions and Condition Context Keys for Amazon Elastic File System \(p. 535\)](#)
- [Actions and Condition Context Keys for Elastic Load Balancing \(p. 536\)](#)
- [Actions and Condition Context Keys for Amazon Elastic MapReduce \(p. 538\)](#)
- [Actions and Condition Context Keys for Amazon Elastic Transcoder \(p. 538\)](#)
- [Actions and Condition Context Keys for Amazon ElastiCache \(p. 539\)](#)
- [Actions and Condition Context Keys for Amazon Elasticsearch Service \(p. 540\)](#)
- [Actions and Condition Context Keys for AWS Elemental MediaConvert \(p. 541\)](#)
- [Actions and Condition Context Keys for AWS Elemental MediaLive \(p. 542\)](#)
- [Actions and Condition Context Keys for AWS Elemental MediaPackage \(p. 542\)](#)
- [Actions and Condition Context Keys for AWS Elemental MediaStore \(p. 543\)](#)
- [Actions and Condition Context Keys for Amazon FreeRTOS \(p. 543\)](#)
- [Actions and Condition Context Keys for Amazon GameLift \(p. 544\)](#)
- [Actions and Condition Context Keys for Amazon Glacier \(p. 545\)](#)
- [Actions and Condition Context Keys for AWS Glue \(p. 546\)](#)
- [Actions and Condition Context Keys for AWS Greengrass \(p. 548\)](#)
- [Actions and Condition Context Keys for Amazon GuardDuty \(p. 550\)](#)
- [Actions and Condition Context Keys for AWS Health APIs and Notifications \(p. 551\)](#)
- [Actions and Condition Context Keys for Identity And Access Management \(p. 552\)](#)
- [Actions and Condition Context Keys for AWS Import Export Disk Service \(p. 555\)](#)
- [Actions and Condition Context Keys for Amazon Inspector \(p. 555\)](#)
- [Actions and Condition Context Keys for AWS IoT \(p. 556\)](#)
- [Actions and Condition Context Keys for AWS IoT Analytics \(p. 559\)](#)
- [Actions and Condition Context Keys for AWS Key Management Service \(p. 560\)](#)

- [Actions and Condition Context Keys for Amazon Kinesis \(p. 561\)](#)
- [Actions and Condition Context Keys for Amazon Kinesis Analytics \(p. 562\)](#)
- [Actions and Condition Context Keys for Amazon Kinesis Firehose \(p. 562\)](#)
- [Actions and Condition Context Keys for Amazon Kinesis Video Streams \(p. 563\)](#)
- [Actions and Condition Context Keys for AWS Lambda \(p. 563\)](#)
- [Actions and Condition Context Keys for Amazon Lex \(p. 564\)](#)
- [Actions and Condition Context Keys for Amazon Lightsail \(p. 565\)](#)
- [Actions and Condition Context Keys for Amazon Machine Learning \(p. 567\)](#)
- [Actions and Condition Context Keys for Manage Amazon API Gateway \(p. 568\)](#)
- [Actions and Condition Context Keys for AWS Marketplace \(p. 568\)](#)
- [Actions and Condition Context Keys for AWS Marketplace Management Portal \(p. 569\)](#)
- [Actions and Condition Context Keys for Amazon Mechanical Turk \(p. 569\)](#)
- [Actions and Condition Context Keys for Amazon Mechanical Turk Crowd \(p. 570\)](#)
- [Actions and Condition Context Keys for Amazon Message Delivery Service \(p. 571\)](#)
- [Actions and Condition Context Keys for AWS Migration Hub \(p. 571\)](#)
- [Actions and Condition Context Keys for Amazon Mobile Analytics \(p. 572\)](#)
- [Actions and Condition Context Keys for AWS Mobile Hub \(p. 572\)](#)
- [Actions and Condition Context Keys for Amazon MQ \(p. 573\)](#)
- [Actions and Condition Context Keys for AWS OpsWorks \(p. 573\)](#)
- [Actions and Condition Context Keys for AWS OpsWorks Configuration Management \(p. 575\)](#)
- [Actions and Condition Context Keys for AWS Organizations \(p. 576\)](#)
- [Actions and Condition Context Keys for Amazon Pinpoint \(p. 577\)](#)
- [Actions and Condition Context Keys for Amazon Polly \(p. 578\)](#)
- [Actions and Condition Context Keys for AWS Price List \(p. 578\)](#)
- [Actions and Condition Context Keys for Amazon RDS \(p. 579\)](#)
- [Actions and Condition Context Keys for Amazon Redshift \(p. 581\)](#)
- [Actions and Condition Context Keys for Amazon Rekognition \(p. 583\)](#)
- [Actions and Condition Context Keys for Amazon Resource Group Tagging API \(p. 584\)](#)
- [Actions and Condition Context Keys for AWS Resource Groups \(p. 585\)](#)
- [Actions and Condition Context Keys for Amazon Route 53 \(p. 585\)](#)
- [Actions and Condition Context Keys for Amazon Route 53 Auto Naming \(p. 587\)](#)
- [Actions and Condition Context Keys for Amazon Route53 Domains \(p. 587\)](#)
- [Actions and Condition Context Keys for Amazon S3 \(p. 588\)](#)
- [Actions and Condition Context Keys for Amazon SageMaker \(p. 591\)](#)
- [Actions and Condition Context Keys for AWS Security Token Service \(p. 592\)](#)
- [Actions and Condition Context Keys for AWS Service Catalog \(p. 592\)](#)
- [Actions and Condition Context Keys for Amazon SES \(p. 594\)](#)
- [Actions and Condition Context Keys for AWS Shield \(p. 596\)](#)
- [Actions and Condition Context Keys for Amazon Simple Systems Manager \(p. 596\)](#)
- [Actions and Condition Context Keys for Amazon Simple Workflow Service \(p. 599\)](#)
- [Actions and Condition Context Keys for Amazon SimpleDB \(p. 600\)](#)
- [Actions and Condition Context Keys for Single Sign-On \(p. 601\)](#)
- [Actions and Condition Context Keys for AWS Snowball \(p. 602\)](#)
- [Actions and Condition Context Keys for Amazon SNS \(p. 603\)](#)
- [Actions and Condition Context Keys for Amazon SQS \(p. 604\)](#)

- [Actions and Condition Context Keys for AWS Step Functions \(p. 605\)](#)
- [Actions and Condition Context Keys for Amazon Storage Gateway \(p. 605\)](#)
- [Actions and Condition Context Keys for AWS Support \(p. 607\)](#)
- [Actions and Condition Context Keys for Amazon Translate \(p. 608\)](#)
- [Actions and Condition Context Keys for AWS Trusted Advisor \(p. 608\)](#)
- [Actions and Condition Context Keys for AWS WAF \(p. 609\)](#)
- [Actions and Condition Context Keys for AWS WAF Regional \(p. 610\)](#)
- [Actions and Condition Context Keys for Amazon WorkDocs \(p. 612\)](#)
- [Actions and Condition Context Keys for Amazon WorkMail \(p. 613\)](#)
- [Actions and Condition Context Keys for Amazon WorkSpaces \(p. 614\)](#)
- [Actions and Condition Context Keys for AWS XRay \(p. 615\)](#)

Actions and Condition Context Keys for Alexa for Business

Alexa for Business (service prefix: a4b) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Alexa for Business

- `a4b:SearchProfiles`
- `a4b:DeleteSkillGroup`
- `a4b:UpdateProfile`
- `a4b:TagResource`
- `a4b:UpdateDevice`
- `a4b:ResolveRoom`
- `a4b:DeleteUser`
- `a4b:UpdateRoom`
- `a4b:UpdateSkillGroup`
- `a4b:GetRoom`
- `a4b:AssociateDeviceWithRoom`
- `a4b:DisassociateSkillGroupFromRoom`
- `a4b:SearchSkillGroups`
- `a4b:StartDeviceSync`
- `a4b:PutRoomSkillParameter`
- `a4b:GetDevice`
- `a4b:SearchUsers`
- `a4b:GetSkillGroup`
- `a4b:SearchRooms`
- `a4b:GetRoomSkillParameter`
- `a4b:DeleteProfile`
- `a4b:CreateUser`
- `a4b:AssociateSkillGroupWithRoom`
- `a4b>ListSkills`
- `a4b:DeleteRoomSkillParameter`
- `a4b:UntagResource`
- `a4b:SendInvitation`

- `a4b>ListTags`
- `a4b>DeleteRoom`
- `a4b>CreateProfile`
- `a4b>CreateRoom`
- `a4b/SearchDevices`
- `a4b/CreateSkillGroup`
- `a4b/DisassociateDeviceFromRoom`
- `a4b:GetProfile`
- `a4b/RevokeInvitation`

Condition context keys for Alexa for Business

Alexa for Business has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon API Gateway

Amazon API Gateway (service prefix: `execute-api`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon API Gateway

For more information about controlling access to API Gateway, see [User Access Permissions for Amazon API Gateway](#) in the *API Gateway Developer Guide*.

- `execute-api:InvalidateCache`
- `execute-api:Invoke`

Condition context keys for Amazon API Gateway

Amazon API Gateway has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Application Auto Scaling

Application Auto Scaling (service prefix: `application-autoscaling`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Application Auto Scaling

- `application-autoscaling>DeleteScheduledAction`
- `application-autoscaling:DescribeScheduledActions`
- `application-autoscaling:DescribeScalingActivities`
- `application-autoscaling:RegisterScalableTarget`
- `application-autoscaling:DescribeScalableTargets`
- `application-autoscaling>DeleteScalingPolicy`
- `application-autoscaling:DescribeScalingPolicies`
- `application-autoscaling:DeregisterScalableTarget`
- `application-autoscaling:PutScalingPolicy`

- `application-autoscaling:PutScheduledAction`

Condition context keys for Application Auto Scaling

Application Auto Scaling has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Application Discovery

Application Discovery (service prefix: discovery) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Application Discovery

For information about using the following Application Discovery Service actions in an IAM policy, see [Setting Up Access to Application Discovery Service](#) in the *Application Discovery Service User Guide*.

- `discovery:StartDataCollectionByAgentIds`
- `discovery:DescribeExportConfigurations`
- `discovery:DescribeTags`
- `discovery:StopDataCollectionByAgentIds`
- `discovery:GetDiscoverySummary`
- `discovery:ExportConfigurations`
- `discovery>ListServerNeighbors`
- `discovery:DescribeConfigurations`
- `discovery>CreateApplication`
- `discovery>ListConfigurations`
- `discovery:StartExportTask`
- `discovery>DeleteApplications`
- `discovery:DescribeAgents`
- `discovery:AssociateConfigurationItemsToApplication`
- `discovery>DeleteTags`
- `discovery>CreateTags`
- `discovery:DisassociateConfigurationItemsFromApplication`
- `discovery:UpdateApplication`

Condition context keys for Application Discovery

Application Discovery has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon AppStream

Amazon AppStream (service prefix: appstream) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon AppStream

For more information about the Amazon AppStream API actions in the following list, see [Amazon AppStream REST API](#) in the *Amazon AppStream Developer Guide*.

- `appstream>DeleteImage`
- `appstream>ListAssociatedFleets`
- `appstream>StartFleet`
- `appstream>TagResource`
- `appstream>CreateFleet`
- `appstream>ExpireSession`
- `appstream>StartImageBuilder`
- `appstream>DescribeFleets`
- `appstream>CreateStreamingURL`
- `appstream>UpdateStack`
- `appstream>DescribeSessions`
- `appstream>AssociateFleet`
- `appstream>CreateImageBuilder`
- `appstream>CreateStack`
- `appstream>Stream`
- `appstream>DisassociateFleet`
- `appstream>DeleteStack`
- `appstream>DeleteImageBuilder`
- `appstream>DescribeImages`
- `appstream>CreateImageBuilderStreamingURL`
- `appstream>ListAssociatedStacks`
- `appstream>UpdateDirectoryConfig`
- `appstream>DescribeImageBuilders`
- `appstream>DeleteFleet`
- `appstream>DescribeStacks`
- `appstream>StopImageBuilder`
- `appstream>UpdateFleet`
- `appstream>UntagResource`
- `appstream>DescribeDirectoryConfigs`
- `appstream>StopFleet`
- `appstream>DeleteDirectoryConfig`
- `appstream>CreateDirectoryConfig`
- `appstream>ListTagsForResource`

Condition context keys for Amazon AppStream

Amazon AppStream has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `appstream:userId`

Actions and Condition Context Keys for AWS Artifact

AWS Artifact (service prefix: `artifact`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Artifact

- `artifact:TerminateAgreement`
- `artifact:Get`
- `artifact:DownloadAgreement`
- `artifact:AcceptAgreement`

Condition context keys for AWS Artifact

AWS Artifact has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Athena

Amazon Athena (service prefix: `athena`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Athena

For information about using the following Athena API actions in an IAM policy, see [Setting User and Amazon S3 Bucket Permissions](#) in the *Amazon Athena User Guide*.

- `athena:GetCatalogs`
- `athena:BatchGetNamedQuery`
- `athena:StartQueryExecution`
- `athena:RunQuery`
- `athena:GetNamedQuery`
- `athena:GetNamespaces`
- `athena:CancelQueryExecution`
- `athena:GetTable`
- `athena:GetQueryExecutions`
- `athena:StopQueryExecution`
- `athena:GetExecutionEngines`
- `athena:BatchGetQueryExecution`
- `athena:GetQueryExecution`
- `athena:GetExecutionEngine`
- `athena:GetNamespace`
- `athena:DeleteNamedQuery`
- `athena:GetTables`
- `athena>CreateNamedQuery`
- `athena:GetQueryResults`
- `athena>ListQueryExecutions`
- `athena>ListNamedQueries`

Condition context keys for Amazon Athena

Amazon Athena has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Auto Scaling

Auto Scaling (service prefix: `autoscaling`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Auto Scaling

For information about using the following Auto Scaling API actions in an IAM policy, see [Auto Scaling Actions](#) in the *Amazon EC2 Auto Scaling User Guide*.

- `autoscaling:EnterStandby`
- `autoscaling:PutNotificationConfiguration`
- `autoscaling:DeleteLaunchConfiguration`
- `autoscaling>CreateOrUpdateTags`
- `autoscaling:PutScheduledUpdateGroupAction`
- `autoscaling:AttachLoadBalancerTargetGroups`
- `autoscaling:SetDesiredCapacity`
- `autoscaling:DeleteLifecycleHook`
- `autoscaling:DeleteNotificationConfiguration`
- `autoscaling:PutScalingPolicy`
- `autoscaling:DescribeAutoScalingGroups`
- `autoscaling:DescribeLifecycleHooks`
- `autoscaling:DescribeTags`
- `autoscaling:DescribeAdjustmentTypes`
- `autoscaling:DescribeAutoScalingNotificationTypes`
- `autoscaling:DescribeLoadBalancerTargetGroups`
- `autoscaling:DescribeLoadBalancers`
- `autoscaling:UpdateAutoScalingGroup`
- `autoscaling:DeleteScheduledAction`
- `autoscaling:DetachLoadBalancerTargetGroups`
- `autoscaling:AttachInstances`
- `autoscaling:TerminateInstanceInAutoScalingGroup`
- `autoscaling:ExitStandby`
- `autoscaling:DescribeScalingActivities`
- `autoscaling:DescribeNotificationConfigurations`
- `autoscaling:DetachLoadBalancers`
- `autoscaling:ExecutePolicy`
- `autoscaling:DeletePolicy`
- `autoscaling:DescribePolicies`
- `autoscaling:DescribeTerminationPolicyTypes`
- `autoscaling:RecordLifecycleActionHeartbeat`
- `autoscaling:DescribeScheduledActions`
- `autoscaling:DescribeMetricCollectionTypes`
- `autoscaling:DescribeAccountLimits`
- `autoscaling:PutLifecycleHook`
- `autoscaling:DescribeLifecycleHookTypes`
- `autoscaling:AttachLoadBalancers`

- `autoscaling:CompleteLifecycleAction`
- `autoscaling:SuspendProcesses`
- `autoscaling:SetInstanceProtection`
- `autoscaling:DetachInstances`
- `autoscaling>CreateAutoScalingGroup`
- `autoscaling>CreateLaunchConfiguration`
- `autoscaling:EnableMetricsCollection`
- `autoscaling:DisableMetricsCollection`
- `autoscaling:DescribeLaunchConfigurations`
- `autoscaling:ResumeProcesses`
- `autoscaling>DeleteAutoScalingGroup`
- `autoscaling:SetInstanceHealth`
- `autoscaling:DescribeScalingProcessTypes`
- `autoscaling:DeleteTags`
- `autoscaling:DescribeAutoScalingInstances`

Condition context keys for Auto Scaling

For more information about using condition keys in an IAM policy for Auto Scaling, see [Auto Scaling Keys in the Amazon EC2 Auto Scaling User Guide](#).

Auto Scaling has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `autoscaling:ImageId`
- `autoscaling:InstanceType`
- `autoscaling:LaunchConfigurationName`
- `autoscaling:LoadBalancerNames`
- `autoscaling:MaxSize`
- `autoscaling:MinSize`
- `autoscaling:ResourceTag/`
- `autoscaling:SpotPrice`
- `autoscaling:TargetGroupARNs`
- `autoscaling:VPCZoneIdentifiers`
- `aws:RequestTag/`

Actions and Condition Context Keys for Auto Scaling Plans

Auto Scaling Plans (service prefix: `autoscaling-plans`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Auto Scaling Plans

- `autoscaling-plans:DescribeScalingPlanResources`
- `autoscaling-plans:DescribeScalingPlans`
- `autoscaling-plans>CreateScalingPlan`
- `autoscaling-plans>DeleteScalingPlan`

Condition context keys for Auto Scaling Plans

Auto Scaling Plans has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Batch

AWS Batch (service prefix: batch) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Batch

For more information about controlling access to your AWS Batch resources, see [AWS Batch IAM Policies, Roles, and Permissions](#) in the *AWS Batch User Guide*.

- [batch:TerminateJob](#)
- [batch:DescribeJobs](#)
- [batch:UpdateComputeEnvironment](#)
- [batch:UpdateJobQueue](#)
- [batch:CreateComputeEnvironment](#)
- [batch:RegisterJobDefinition](#)
- [batch>CreateJobQueue](#)
- [batch:DescribeJobQueues](#)
- [batch:CancelJob](#)
- [batch:DescribeJobDefinitions](#)
- [batch:DescribeComputeEnvironments](#)
- [batch:SubmitJob](#)
- [batch>DeleteComputeEnvironment](#)
- [batch>ListJobs](#)
- [batch:DeregisterJobDefinition](#)
- [batch>DeleteJobQueue](#)

Condition context keys for AWS Batch

AWS Batch has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Billing

AWS Billing (service prefix: aws-portal) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Billing

For information about controlling access to your billing data by using an IAM policy, see [Billing and Cost Management Permissions Reference](#) in the *AWS Billing and Cost Management User Guide*.

- [aws-portal:ModifyAccount](#)
- [aws-portal:ViewAccount](#)
- [aws-portal:ModifyBilling](#)
- [aws-portal:ViewPaymentMethods](#)

- `aws-portal:ModifyPaymentMethods`
- `aws-portal:ViewBilling`
- `aws-portal:ViewUsage`

Condition context keys for AWS Billing

AWS Billing has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Budget Service

AWS Budget Service (service prefix: budgets) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Budget Service

For more information about controlling access to AWS Billing, see [Billing and Cost Management Permissions Reference](#) in the *AWS Billing and Cost Management API Reference*.

- `budgets:ViewBudget`
- `budgets:ModifyBudget`

Condition context keys for AWS Budget Service

AWS Budget Service has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Certificate Manager

AWS Certificate Manager (service prefix: acm) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Certificate Manager

For information about controlling access to ACM by using an IAM policy, see [Permissions and Policies](#) in the *AWS Certificate Manager User Guide*.

- `acm:DescribeCertificate`
- `acm:ResendValidationEmail`
- `acm:RequestCertificate`
- `acm:DeleteCertificate`
- `acm:RemoveTagsFromCertificate`
- `acm:ImportCertificate`
- `acm:AddTagsToCertificate`
- `acm>ListTagsForCertificate`
- `acm>ListCertificates`
- `acm:GetCertificate`

Condition context keys for AWS Certificate Manager

AWS Certificate Manager has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Chime

Amazon Chime (service prefix: chime) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Chime

For information about using the following Amazon Chime actions in an IAM policy, see [Control Access to the Amazon Chime Console](#) in the *Amazon Chime Administration Guide*.

- chime:RenameAccount
- chime>ListUsers
- chime:ResetPersonalPin
- chime>ListAccounts
- chime>DeleteCDRBucket
- chime:ConnectDirectory
- chime>ListGroups
- chime:UpdateAccountResource
- chime:LogoutUser
- chime>DeleteDelegate
- chime:ActivateUsers
- chime:ResetAccountResource
- chime:SuspendUsers
- chime:UpdateAccountSettings
- chime:GetCDRBucket
- chime>DeleteGroups
- chime:GetAccountResource
- chime>ListDomains
- chime>ListDirectories
- chime>DeleteAccount
- chime:RenewDelegate
- chime:GetUser
- chime:GetUserByEmail
- chime>ListCDRBucket
- chime:UpdateCDRBucket
- chime:InviteUsers
- chime:ValidateAccountResource
- chime:GetAccount
- chime:AddOrUpdateGroups
- chime:GetDomain
- chime>CreateAccount
- chime:GetAccountSettings
- chime:SubmitSupportRequest
- chime:ValidateDelegate

- chime>ListDelegates
- chime>AcceptDelegate
- chime>UpdateUserLicenses
- chime>DeleteDomain
- chime>AuthorizeDirectory
- chime>AddDomain
- chime>UpdateSupportedLicenses
- chime>CreateCDRBucket
- chime>InviteDelegate
- chime>DisconnectDirectory
- chime>UnauthorizeDirectory

Condition context keys for Amazon Chime

Amazon Chime has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon AWS Cloud Contact Center

Amazon AWS Cloud Contact Center (service prefix: connect) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon AWS Cloud Contact Center

For information about using Amazon Connect, see [Amazon Connect Documentation](#).

- connect>GetFederationTokens
- connect>DescribeInstance
- connect>DestroyInstance
- connect>ListInstances
- connect>CreateInstance
- connect>ModifyInstance
- connect>GetFederationToken

Condition context keys for Amazon AWS Cloud Contact Center

Amazon AWS Cloud Contact Center has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Cloud Directory

Amazon Cloud Directory (service prefix: clouddirectory) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Cloud Directory

For additional information about using Amazon Cloud Directory actions in an IAM policy, see [Amazon Cloud Directory API Permissions: Actions, Resources, and Conditions Reference](#) in the *Amazon Cloud Directory Administration Guide*.

- [clouddirectory:BatchWrite](#)
- [clouddirectory>ListDevelopmentSchemaArns](#)
- [clouddirectory:UpdateTypedLinkFacet](#)
- [clouddirectory:DisableDirectory](#)
- [clouddirectory:GetSchemaAsJson](#)
- [clouddirectory>ListTypedLinkFacetAttributes](#)
- [clouddirectory>ListPublishedSchemaArns](#)
- [clouddirectory>DeleteFacet](#)
- [clouddirectory:GetDirectory](#)
- [clouddirectory>CreateTypedLinkFacet](#)
- [clouddirectory>ListAppliedSchemaArns](#)
- [clouddirectory:GetTypedLinkFacetInformation](#)
- [clouddirectory:DetachFromIndex](#)
- [clouddirectory:EnableDirectory](#)
- [clouddirectory:AttachPolicy](#)
- [clouddirectory:UpdateFacet](#)
- [clouddirectory>ListObjectAttributes](#)
- [clouddirectory>ListObjectPolicies](#)
- [clouddirectory:LookupPolicy](#)
- [clouddirectory:AttachToIndex](#)
- [clouddirectory:UpdateSchema](#)
- [clouddirectory>DeleteDirectory](#)
- [clouddirectory>ListTypedLinkFacetNames](#)
- [clouddirectory:ApplySchema](#)
- [clouddirectory>ListOutgoingTypedLinks](#)
- [clouddirectory:UntagResource](#)
- [clouddirectory>CreateSchema](#)
- [clouddirectory:PutSchemaFromJson](#)
- [clouddirectory>ListTagsForResource](#)
- [clouddirectory>DeleteTypedLinkFacet](#)
- [clouddirectory:PublishSchema](#)
- [clouddirectory>ListObjectParentPaths](#)
- [clouddirectory>ListObjectChildren](#)
- [clouddirectory>DeleteObject](#)
- [clouddirectory>DeleteSchema](#)
- [clouddirectory>ListAttachedIndices](#)
- [clouddirectory:TagResource](#)
- [clouddirectory:DetachTypedLink](#)
- [clouddirectory:BatchRead](#)
- [clouddirectory:DetachPolicy](#)
- [clouddirectory>ListPolicyAttachments](#)
- [clouddirectory>CreateIndex](#)
- [clouddirectory:AttachTypedLink](#)
- [clouddirectory:RemoveFacetFromObject](#)
- [clouddirectory>ListDirectories](#)

- `clouddirectory>ListIncomingTypedLinks`
- `clouddirectory:GetFacet`
- `clouddirectory:GetObjectInformation`
- `clouddirectory:UpdateObjectAttributes`
- `clouddirectory>ListFacetAttributes`
- `clouddirectory>CreateFacet`
- `clouddirectory>CreateDirectory`
- `clouddirectoryDetachObject`
- `clouddirectory>ListFacetNames`
- `clouddirectory>ListObjectParents`
- `clouddirectoryAttachObject`
- `clouddirectoryListItemIcon`
- `clouddirectoryCreateObject`
- `clouddirectoryAddFacetToObject`

Condition context keys for Amazon Cloud Directory

Amazon Cloud Directory has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Cloud9

AWS Cloud9 (service prefix: cloud9) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Cloud9

- `cloud9:DescribeEnvironments`
- `cloud9>ListEnvironments`
- `cloud9>CreateEnvironmentEC2`
- `cloud9:UpdateEnvironmentMembership`
- `cloud9: GetUserPublicKey`
- `cloud9>CreateEnvironmentSSH`
- `cloud9>DeleteEnvironmentMembership`
- `cloud9:DescribeEnvironmentStatus`
- `cloud9:DeleteEnvironment`
- `cloud9:DescribeEnvironmentMemberships`
- `cloud9:CreateEnvironmentMembership`
- `cloud9:ValidateEnvironmentName`
- `cloud9:UpdateEnvironment`

Condition context keys for AWS Cloud9

AWS Cloud9 has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `cloud9:InstanceType`

- `cloud9:SubnetId`
- `cloud9:UserArn`
- `cloud9:EnvironmentId`
- `cloud9:EnvironmentName`
- `cloud9:Permissions`

Actions and Condition Context Keys for AWS CloudFormation

AWS CloudFormation (service prefix: `cloudformation`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS CloudFormation

For information about using the following AWS CloudFormation API actions in an IAM policy, see [AWS CloudFormation Actions and Resources](#) in the *AWS CloudFormation User Guide*.

- `cloudformation>ListImports`
- `cloudformation>ListStacks`
- `cloudformation>DescribeChangeSet`
- `cloudformation>ValidateTemplate`
- `cloudformation>UpdateStack`
- `cloudformation>GetTemplateSummary`
- `cloudformation>DescribeAccountLimits`
- `cloudformation>CreateStack`
- `cloudformation>DescribeStackResource`
- `cloudformation>DescribeStackEvents`
- `cloudformation>ListExports`
- `cloudformation>PreviewStackUpdate`
- `cloudformation>DeleteStack`
- `cloudformation>CreateUploadBucket`
- `cloudformation>DeleteChangeSet`
- `cloudformation>SignalResource`
- `cloudformation>ExecuteChangeSet`
- `cloudformation>DescribeStacks`
- `cloudformation>CreateChangeSet`
- `cloudformation>SetStackPolicy`
- `cloudformation>GetTemplate`
- `cloudformation>DescribeStackResources`
- `cloudformation>GetStackPolicy`
- `cloudformation>UpdateTerminationProtection`
- `cloudformation>EstimateTemplateCost`
- `cloudformation>CancelUpdateStack`
- `cloudformation>ContinueUpdateRollback`
- `cloudformation>ListStackResources`
- `cloudformation>ListChangeSets`

Condition context keys for AWS CloudFormation

For information about using conditions in an IAM policy, see [AWS CloudFormation Conditions in the AWS CloudFormation User Guide](#).

AWS CloudFormation has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `cloudformation:ChangeSetName`
- `cloudformation:ResourceTypes`
- `cloudformation:RoleArn`
- `cloudformation:StackPolicyUrl`
- `cloudformation:TemplateUrl`

Actions and Condition Context Keys for Amazon CloudFront

Amazon CloudFront (service prefix: `cloudfront`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon CloudFront

For information about using the following CloudFront API actions in an IAM policy, see [CloudFront Actions](#) in the *Amazon CloudFront Developer Guide*.

- `cloudfront>ListStreamingDistributions`
- `cloudfront>CreateStreamingDistribution`
- `cloudfront>UpdateStreamingDistribution`
- `cloudfront>TagResource`
- `cloudfront>CreateDistribution`
- `cloudfront>DeleteDistribution`
- `cloudfront>CreateInvalidation`
- `cloudfront>DeleteStreamingDistribution`
- `cloudfront>GetDistributionConfig`
- `cloudfront>ListDistributionsByWebACLId`
- `cloudfront>GetStreamingDistributionConfig`
- `cloudfront>GetDistribution`
- `cloudfront>ListDistributions`
- `cloudfront>ListCloudFrontOriginAccessIdentities`
- `cloudfront>DeleteCloudFrontOriginAccessIdentity`
- `cloudfront>CreateCloudFrontOriginAccessIdentity`
- `cloudfront>GetCloudFrontOriginAccessIdentityConfig`
- `cloudfront>UpdateDistribution`
- `cloudfront>UpdateCloudFrontOriginAccessIdentity`
- `cloudfront>GetStreamingDistribution`
- `cloudfront>CreateDistributionWithTags`
- `cloudfront>ListInvalidations`
- `cloudfront>GetCloudFrontOriginAccessIdentity`
- `cloudfront>UntagResource`
- `cloudfront>GetInvalidation`
- `cloudfront>CreateStreamingDistributionWithTags`

- [cloudfront>ListTagsForResource](#)

Condition context keys for Amazon CloudFront

For information about using condition keys in an IAM policy, see [Policy Keys](#) in the *Amazon CloudFront Developer Guide*.

Amazon CloudFront has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS CloudHSM

AWS CloudHSM (service prefix: clouhdsm) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS CloudHSM

For information about controlling access to AWS CloudHSM by using an IAM policy, see [Controlling Access to AWS CloudHSM Resources](#) in the *AWS CloudHSM User Guide*.

- [clouhdsm>ModifyHsm](#)
- [clouhdsm>DeleteHapg](#)
- [clouhdsm>ListAvailableZones](#)
- [clouhdsm>CreateHapg](#)
- [clouhdsm>DescribeHapg](#)
- [clouhdsm>ListHsms](#)
- [clouhdsm>DescribeHsm](#)
- [clouhdsm>DeleteHsm](#)
- [clouhdsm>ListLunaClients](#)
- [clouhdsm>DeleteLunaClient](#)
- [clouhdsm>CreateLunaClient](#)
- [clouhdsm>ListHapgs](#)
- [clouhdsm>ModifyHapg](#)
- [clouhdsm>RemoveTagsFromResource](#)
- [clouhdsm>ModifyLunaClient](#)
- [clouhdsm>GetConfig](#)
- [clouhdsm>CreateHsm](#)
- [clouhdsm>AddTagsToResource](#)
- [clouhdsm>DescribeLunaClient](#)
- [clouhdsm>ListTagsForResource](#)

Condition context keys for AWS CloudHSM

AWS CloudHSM has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon CloudSearch

Amazon CloudSearch (service prefix: cloudsearch) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon CloudSearch

For additional information about using Amazon CloudSearch actions in an IAM policy, see [Configuring Access for Amazon CloudSearch](#) in the *Amazon CloudSearch Developer Guide*.

- `cloudsearch:DescribeScalingParameters`
- `cloudsearch:document` - this is an IAM policy permission only, not an API action that can be called.
- `cloudsearch:DefineSuggester`
- `cloudsearch:RemoveTags`
- `cloudsearch:UpdateAvailabilityOptions`
- `cloudsearch:search` - this is an IAM policy permission only, not an API action that can be called.
- `cloudsearch:DescribeDomains`
- `cloudsearch:DeleteAnalysisScheme`
- `cloudsearch:UpdateScalingParameters`
- `cloudsearch>ListDomainNames`
- `cloudsearch:DeleteExpression`
- `cloudsearch:DefineExpression`
- `cloudsearch:DescribeAvailabilityOptions`
- `cloudsearch:IndexDocuments`
- `cloudsearch:DescribeIndexFields`
- `cloudsearch:DeleteSuggester`
- `cloudsearch:DescribeExpressions`
- `cloudsearch:DescribeServiceAccessPolicies`
- `cloudsearch:suggest` - this is an IAM policy permission only, not an API action that can be called.
- `cloudsearch:DescribeAnalysisSchemes`
- `cloudsearch:DeleteIndexField`
- `cloudsearch:DefineIndexField`
- `cloudsearch:DeleteDomain`
- `cloudsearch>CreateDomain`
- `cloudsearch:DescribeSuggesters`
- `cloudsearch:DefineAnalysisScheme`
- `cloudsearch:AddTags`
- `cloudsearch>ListTags`
- `cloudsearch:BuildSuggesters`
- `cloudsearch:UpdateServiceAccessPolicies`

Condition context keys for Amazon CloudSearch

Amazon CloudSearch has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS CloudTrail

AWS CloudTrail (service prefix: `clouptrail`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS CloudTrail

For more information about using the following CloudTrail API actions in an IAM policy, see [Granting Permissions for CloudTrail Actions](#) in the AWS CloudTrail User Guide.

- [cloudtrail:StopLogging](#)
- [cloudtrail:UpdateTrail](#)
- [cloudtrail>CreateTrail](#)
- [cloudtrail:DescribeTrails](#)
- [cloudtrail:PutEventSelectors](#)
- [cloudtrail:GetTrailStatus](#)
- [cloudtrail>ListPublicKeys](#)
- [cloudtrail:RemoveTags](#)
- [cloudtrail:StartLogging](#)
- [cloudtrail>DeleteTrail](#)
- [cloudtrail>ListTags](#)
- [cloudtrail>AddTags](#)
- [cloudtrail:LookupEvents](#)
- [cloudtrail:GetEventSelectors](#)

Condition context keys for AWS CloudTrail

AWS CloudTrail has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon CloudWatch

Amazon CloudWatch (service prefix: cloudwatch) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon CloudWatch

For information about using the following CloudWatch API actions in an IAM policy, see [CloudWatch Actions](#) in the *Amazon CloudWatch User Guide*.

- [cloudwatch:PutMetricData](#)
- [cloudwatch:PutMetricAlarm](#)
- [cloudwatch:GetMetricData](#) - this is an IAM policy permission only, not an API action that can be called.
- [cloudwatch>DeleteAlarms](#)
- [cloudwatch:GetMetricStatistics](#)
- [cloudwatch:SetAlarmState](#)
- [cloudwatch:DescribeAlarmsForMetric](#)
- [cloudwatch:DescribeAlarmHistory](#)
- [cloudwatch>ListMetrics](#)
- [cloudwatch:EnableAlarmActions](#)
- [cloudwatch:DisableAlarmActions](#)
- [cloudwatch:DescribeAlarms](#)

Condition context keys for Amazon CloudWatch

For information about using conditions to control access to CloudWatch in an IAM policy, see [CloudWatch Keys](#) in the *Amazon CloudWatch User Guide*.

Amazon CloudWatch has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon CloudWatch Events

Amazon CloudWatch Events (service prefix: events) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon CloudWatch Events

For information about controlling access to CloudWatch Events by using an IAM policy, see [Controlling User Access to Rules and Events](#) in the *Amazon CloudWatch User Guide*.

- `events:PutEvents`
- `events:DescribeRule`
- `events:EnableRule`
- `events:DisableRule`
- `events:PutRule`
- `events:DeleteRule`
- `events:PutTargets`
- `events>ListRules`
- `events:TestEventPattern`
- `events:RemoveTargets`
- `events>ListTargetsByRule`
- `events>ListRuleNamesByTarget`

Condition context keys for Amazon CloudWatch Events

For information about using condition keys to provide more granular control over CloudWatch Events with IAM policies, see [Condition Keys for CloudWatch Events](#) in the *Amazon CloudWatch User Guide*.

Amazon CloudWatch Events has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `events:TargetArn`
- `events:detail-type`
- `events:detail.userIdentity.principalId`
- `events:source`

Actions and Condition Context Keys for Amazon CloudWatch Logs

Amazon CloudWatch Logs (service prefix: logs) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon CloudWatch Logs

- logs:PutDestination
- logs:TagLogGroup
- logs>DeleteRetentionPolicy
- logs:PutResourcePolicy
- logs:PutDestinationPolicy
- logs:PutRetentionPolicy
- logs:DescribeExportTasks
- logs:DeleteLogStream
- logs:DescribeLogStreams
- logs:DeleteLogGroup
- logs:PutLogEvents
- logs:UntagLogGroup
- logs:PutMetricFilter
- logs>CreateExportTask
- logs:DescribeSubscriptionFilters
- logs>CreateLogGroup
- logs:DeleteDestination
- logs:DeleteSubscriptionFilter
- logs:DeleteResourcePolicy
- logs>ListTagsLogGroup
- logs:FilterLogEvents
- logs:PutSubscriptionFilter
- logs:DescribeResourcePolicies
- logs:DescribeMetricFilters
- logs:AssociateKmsKey
- logs:GetLogEvents
- logs:CancelExportTask
- logs:TestMetricFilter
- logs>CreateLogStream
- logs:DescribeDestinations
- logs:DescribeLogGroups
- logs:DisassociateKmsKey
- logs:DeleteMetricFilter

Condition context keys for Amazon CloudWatch Logs

Amazon CloudWatch Logs has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Code Signing for Amazon FreeRTOS

AWS Code Signing for Amazon FreeRTOS (service prefix: signer) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Code Signing for Amazon FreeRTOS

- `signer:DescribeSigningJob`
- `signer>ListSigningJobs`
- `signer:StartSigningJob`

Condition context keys for AWS Code Signing for Amazon FreeRTOS

AWS Code Signing for Amazon FreeRTOS has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS CodeBuild

AWS CodeBuild (service prefix: `codebuild`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS CodeBuild

For information about using the following AWS CodeBuild API actions in an IAM policy, see [Using Identity-Based Policies \(IAM Policies\) for AWS CodeBuild](#) in the *AWS CodeBuild User Guide*.

- `codebuild:PersistOAuthToken` - this is an IAM policy permission only, not an API action that can be called.
- `codebuild>ListProjects`
- `codebuild:BatchGetProjects`
- `codebuild>ListConnectedOAuthAccounts` - this is an IAM policy permission only, not an API action that can be called.
- `codebuild>ListCuratedEnvironmentImages`
- `codebuild:BatchDeleteBuilds`
- `codebuild>CreateProject`
- `codebuild:StopBuild`
- `codebuild>ListBuildsForProject`
- `codebuild>ListRepositories` - this is an IAM policy permission only, not an API action that can be called.
- `codebuild>DeleteProject`
- `codebuild:UpdateProject`
- `codebuild>ListBuilds`
- `codebuild:StartBuild`
- `codebuild:BatchGetBuilds`

Condition context keys for AWS CodeBuild

AWS CodeBuild has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS CodeCommit

AWS CodeCommit (service prefix: `codecommit`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS CodeCommit

For information about using the following AWS CodeCommit actions in an IAM policy, see [Access Permissions Reference](#) in the [AWS CodeCommit User Guide](#).

- `codecommit:GetBranch`
- `codecommit:GetBlob`
- `codecommit:PutRepositoryTriggers`
- `codecommit:GetCommit`
- `codecommit:GetRepository`
- `codecommit:GitPull` - this is an IAM policy permission only, not an API action that can be called.
- `codecommit:GetRepositoryTriggers`
- `codecommit:DescribePullRequestEvents`
- `codecommit:GetCommitHistory`
- `codecommit:GetTree` - this is an IAM policy permission only, not an API action that can be called.
- `codecommit:GetComment`
- `codecommit:UpdatePullRequestDescription`
- `codecommit>ListBranches`
- `codecommit>DeleteRepository`
- `codecommit>ListRepositories`
- `codecommit:UpdateRepositoryDescription`
- `codecommit:GetPullRequest`
- `codecommit:UpdateRepositoryName`
- `codecommit:GetUploadArchiveStatus`
- `codecommit:PostCommentReply`
- `codecommit:GetCommitsFromMergeBase`
- `codecommit>DeleteBranch`
- `codecommit>CreateRepository`
- `codecommit>ListPullRequests`
- `codecommit:UpdatePullRequestStatus`
- `codecommit:GetCommentsForPullRequest`
- `codecommit:GetMergeConflicts`
- `codecommit:GetObjectIdentifier` - this is an IAM policy permission only, not an API action that can be called.
- `codecommit>DeleteCommentContent`
- `codecommit:MergePullRequestByFastForward`
- `codecommit>CreatePullRequest`
- `codecommit:TestRepositoryTriggers`
- `codecommit:UpdateComment`
- `codecommit:PostCommentForComparedCommit`
- `codecommit:GetCommentsForComparedCommit`
- `codecommit:UpdateDefaultBranch`
- `codecommit>CreateBranch`
- `codecommit:UpdatePullRequestTitle`
- `codecommit:GetReferences`
- `codecommit:UploadArchive`
- `codecommit:BatchGetPullRequests`
- `codecommit:GitPush` - this is an IAM policy permission only, not an API action that can be called.

- `codecommit:BatchGetRepositories`
- `codecommit:CancelUploadArchive`
- `codecommit:PostCommentForPullRequest`

Condition context keys for AWS CodeCommit

AWS CodeCommit has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS CodeDeploy

AWS CodeDeploy (service prefix: `codedeploy`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS CodeDeploy

For information about using the following AWS CodeDeploy API actions in an IAM policy, see [AWS CodeDeploy User Access Permissions Reference](#) in the *AWS CodeDeploy User Guide*.

- `codedeploy:BatchGetDeployments`
- `codedeploy>ListDeploymentInstances`
- `codedeploy>ListOnPremisesInstances`
- `codedeploy:RemoveTagsFromOnPremisesInstances`
- `codedeploy>CreateDeploymentConfig`
- `codedeploy:BatchGetDeploymentGroups`
- `codedeploy:UpdateDeploymentGroup`
- `codedeploy:GetDeploymentInstance`
- `codedeploy:GetApplicationRevision`
- `codedeploy:RegisterOnPremisesInstance`
- `codedeploy:RegisterApplicationRevision`
- `codedeploy:StopDeployment`
- `codedeploy:BatchGetApplicationRevisions`
- `codedeploy>CreateApplication`
- `codedeploy:GetDeploymentGroup`
- `codedeploy:DeregisterOnPremisesInstance`
- `codedeploy:AddTagsToOnPremisesInstances`
- `codedeploy:UpdateApplication`
- `codedeploy:GetOnPremisesInstance`
- `codedeploy:BatchGetDeploymentInstances`
- `codedeploy:GetDeployment`
- `codedeploy:BatchGetOnPremisesInstances`
- `codedeploy:ContinueDeployment`
- `codedeploy>ListApplicationRevisions`
- `codedeploy:BatchGetApplications`
- `codedeploy>DeleteDeploymentConfig`
- `codedeploy>ListDeployments`
- `codedeploy>ListDeploymentConfigs`
- `codedeploy>CreateDeployment`

- `codedeploy>ListApplications`
- `codedeploy>DeleteApplication`
- `codedeploy>DeleteDeploymentGroup`
- `codedeploy>GetApplication`
- `codedeploy>CreateDeploymentGroup`
- `codedeploy>GetDeploymentConfig`
- `codedeploy>ListDeploymentGroups`

Condition context keys for AWS CodeDeploy

AWS CodeDeploy has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS CodePipeline

AWS CodePipeline (service prefix: codepipeline) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS CodePipeline

For information about controlling access to AWS CodePipeline by using an IAM policy, see [AWS CodePipeline Access Permissions Reference](#) in the *AWS CodePipeline User Guide*.

- `codepipeline>PutJobFailureResult`
- `codepipeline>PutThirdPartyJobFailureResult`
- `codepipeline>PutActionRevision`
- `codepipeline>DeleteCustomActionType`
- `codepipeline>CreatePipeline`
- `codepipeline>ListPipelines`
- `codepipeline>PollForThirdPartyJobs`
- `codepipeline>PollForJobs`
- `codepipeline>DisableStageTransition`
- `codepipeline>StartPipelineExecution`
- `codepipeline>AcknowledgeJob`
- `codepipeline>UpdatePipeline`
- `codepipeline>GetPipelineState`
- `codepipeline>RetryStageExecution`
- `codepipeline>GetJobDetails`
- `codepipeline>ListActionTypes`
- `codepipeline>PutApprovalResult`
- `codepipeline>CreateCustomActionType`
- `codepipeline>AcknowledgeThirdPartyJob`
- `codepipeline>GetPipelineExecution`
- `codepipeline>GetThirdPartyJobDetails`
- `codepipeline>GetPipeline`
- `codepipeline>ListPipelineExecutions`
- `codepipeline>PutThirdPartyJobSuccessResult`
- `codepipeline>PutJobSuccessResult`

- `codepipeline>DeletePipeline`
- `codepipeline:EnableStageTransition`

Condition context keys for AWS CodePipeline

AWS CodePipeline has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS CodeStar

AWS CodeStar (service prefix: codestar) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS CodeStar

For information about using the following AWS CodeStar API actions in an IAM policy, see [AWS CodeStar Access Permissions Reference](#) in the *AWS CodeStar User Guide*.

- `codestar:DisassociateTeamMember`
- `codestar>ListResources`
- `codestar>ListUserProfiles`
- `codestar>ListProjects`
- `codestar:UpdateTeamMember`
- `codestar>CreateProject`
- `codestar:DescribeProject`
- `codestar:GetExtendedAccess`
- `codestar:UpdateUserProfile`
- `codestar>CreateUserProfile`
- `codestar>ListTeamMembers`
- `codestar>DeleteProject`
- `codestar:UpdateProject`
- `codestar:AssociateTeamMember`
- `codestar:PutExtendedAccess`
- `codestar:DescribeUserProfile`
- `codestar>DeleteExtendedAccess`
- `codestar:VerifyServiceRole`
- `codestar>DeleteUserProfile`

Condition context keys for AWS CodeStar

AWS CodeStar has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Cognito Identity

Amazon Cognito Identity (service prefix: cognito-identity) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Cognito Identity

- `cognito-identity:GetIdentityPoolRoles`
- `cognito-identity:SetIdentityPoolRoles`
- `cognito-identity>DeleteIdentityPool`
- `cognito-identity:UpdateIdentityPool`
- `cognito-identity:GetOpenIdTokenForDeveloperIdentity`
- `cognito-identity:GetId`
- `cognito-identity:UnlinkIdentity`
- `cognito-identity>DeleteIdentities`
- `cognito-identity:DescribeIdentity`
- `cognito-identity:MergeDeveloperIdentities`
- `cognito-identity:LookupDeveloperIdentity`
- `cognito-identity:UnlinkDeveloperIdentity`
- `cognito-identity:GetOpenIdToken`
- `cognito-identity:DescribeIdentityPool`
- `cognito-identity>ListIdentityPools`
- `cognito-identity>CreateIdentityPool`
- `cognito-identity>ListIdentities`
- `cognito-identity:GetCredentialsForIdentity`

Condition context keys for Amazon Cognito Identity

Amazon Cognito Identity has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Cognito Sync

Amazon Cognito Sync (service prefix: `cognito-sync`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Cognito Sync

- `cognito-sync:UpdateRecords`
- `cognito-sync:DescribeIdentityPoolUsage`
- `cognito-sync:QueryRecords`
- `cognito-sync:RegisterDevice`
- `cognito-sync:BulkPublish`
- `cognito-sync:UnsubscribeFromDataset`
- `cognito-sync>ListDatasets`
- `cognito-sync:GetIdentityPoolConfiguration`
- `cognito-sync:SetIdentityPoolConfiguration`
- `cognito-sync>ListRecords`
- `cognito-sync:SetCognitoEvents`
- `cognito-sync:SetDatasetConfiguration`
- `cognito-sync:GetBulkPublishDetails`
- `cognito-sync:GetCognitoEvents`
- `cognito-sync:DescribeIdentityUsage`

- `cognito-sync>ListIdentityPoolUsage`
- `cognito-sync>DescribeDataset`
- `cognito-sync>DeleteDataset`
- `cognito-sync>SubscribeToDataset`

Condition context keys for Amazon Cognito Sync

Amazon Cognito Sync has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Cognito User Pools

Amazon Cognito User Pools (service prefix: `cognito-idp`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Cognito User Pools

For more information about Amazon Cognito Your User Pools, see the [Amazon Cognito Your User Pools documentation](#).

- `cognito-idp>UpdateAuthEventFeedback`
- `cognito-idp>VerifyUserAttribute`
- `cognito-idp>AdminListUserAuthEvents`
- `cognito-idp>ListUsersInGroup`
- `cognito-idp>DescribeUserImportJob`
- `cognito-idp>DescribeUserPool`
- `cognito-idp>AdminListDevices`
- `cognito-idp>AdminEnableUser`
- `cognito-idp>CreateUserImportJob`
- `cognito-idp> GetUserAttributeVerificationCode`
- `cognito-idp>ConfirmDevice`
- `cognito-idp> GetUser`
- `cognito-idp>AdminSetUserMFAPreference`
- `cognito-idp>CreateUserPool`
- `cognito-idp>GlobalSignOut`
- `cognito-idp>AdminInitiateAuth`
- `cognito-idp>AdminResetUserPassword`
- `cognito-idp>AdminUserGlobalSignOut`
- `cognito-idp>ConfirmSignUp`
- `cognito-idp>ForgotPassword`
- `cognito-idp> GetUserPoolMfaConfig`
- `cognito-idp>AdminGetDevice`
- `cognito-idp>ForgetDevice`
- `cognito-idp> GetGroup`
- `cognito-idp>AdminDeleteUserAttributes`
- `cognito-idp>AdminUpdateAuthEventFeedback`
- `cognito-idp>AdminForgetDevice`

- `cognito-idp:DescribeUserPoolClient`
- `cognito-idp:DescribeRiskConfiguration`
- `cognito-idp:AdminConfirmSignUp`
- `cognito-idp:SetUserPoolMfaConfig`
- `cognito-idp:AddCustomAttributes`
- `cognito-idp:AdminUpdateUserAttributes`
- `cognito-idp:GetCSVHeader`
- `cognito-idp:ChangePassword`
- `cognito-idp:SignUp`
- `cognito-idp:AdminListGroupsForUser`
- `cognito-idp>ListDevices`
- `cognito-idp:AdminCreateUser`
- `cognito-idp:InitiateAuth`
- `cognito-idp:AdminSetUserSettings`
- `cognito-idp:DeleteUser`
- `cognito-idp>ListGroups`
- `cognito-idp>DeleteGroup`
- `cognito-idp:UpdateUserPool`
- `cognito-idp:ResendConfirmationCode`
- `cognito-idp:StopUserImportJob`
- `cognito-idp:StartUserImportJob`
- `cognito-idp:Admin GetUser`
- `cognito-idp:GetDevice`
- `cognito-idp:SetRiskConfiguration`
- `cognito-idp:AdminRespondToAuthChallenge`
- `cognito-idp:SetUserMFAPreference`
- `cognito-idp:AdminDeleteUser`
- `cognito-idp:AdminRemoveUserFromGroup`
- `cognito-idp>CreateGroup`
- `cognito-idp:UpdateUserPoolClient`
- `cognito-idp>DeleteUserPoolClient`
- `cognito-idp:RespondToAuthChallenge`
- `cognito-idp:AdminUpdateDeviceStatus`
- `cognito-idp:AdminDisableUser`
- `cognito-idp>ListUserImportJobs`
- `cognito-idp>DeleteUserAttributes`
- `cognito-idp:UpdateDeviceStatus`
- `cognito-idp:ConfirmForgotPassword`
- `cognito-idp>DeleteUserPool`
- `cognito-idp>ListUserPoolClients`
- `cognito-idp>CreateUserPoolClient`
- `cognito-idp:SetUserSettings`
- `cognito-idp:UpdateUserAttributes`
- `cognito-idp:UpdateGroup`
- `cognito-idp:AdminAddUserToGroup`

Condition context keys for Amazon Cognito User Pools

Amazon Cognito User Pools has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Comprehend

Amazon Comprehend (service prefix: comprehend) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Comprehend

- `comprehend:BatchDetectKeyPhrases`
- `comprehend:DetectDominantLanguage`
- `comprehend>ListTopicsDetectionJobs`
- `comprehend:DetectEntities`
- `comprehend:BatchDetectEntities`
- `comprehend:DetectKeyPhrases`
- `comprehend:DetectSentiment`
- `comprehend:StartTopicsDetectionJob`
- `comprehend:DescribeTopicsDetectionJob`
- `comprehend:BatchDetectDominantLanguage`
- `comprehend:BatchDetectSentiment`

Condition context keys for Amazon Comprehend

Amazon Comprehend has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Config

AWS Config (service prefix: config) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Config

For information about using the following AWS Config API actions in an IAM policy, see [Recommended IAM Permissions for Using the AWS Config Console and the AWS CLI](#) in the *AWS Config Developer Guide*.

- `config>DeleteEvaluationResults`
- `config:GetResourceConfigHistory`
- `config:DeliverConfigSnapshot`
- `config:DescribeDeliveryChannelStatus`
- `config:PutConfigRule`
- `config:GetResources`
- `config:StartConfigurationRecorder`
- `config:PutDeliveryChannel`
- `config:StartConfigRulesEvaluation`
- `config:DeleteDeliveryChannel`
- `config:DeleteConfigurationRecorder`

- `config:GetComplianceDetailsByResource`
- `config:GetComplianceDetailsByConfigRule`
- `config:DescribeConfigRules`
- `config>ListDiscoveredResources`
- `config:DescribeConfigRuleEvaluationStatus`
- `config:GetTagKeys`
- `config:DescribeDeliveryChannels`
- `config:DeleteConfigRule`
- `config:PutEvaluations`
- `config:DescribeConfigurationRecorderStatus`
- `config:StopConfigurationRecorder`
- `config:DescribeComplianceByResource`
- `config:DescribeComplianceByConfigRule`
- `config:DescribeConfigurationRecorders`
- `config:GetComplianceSummaryByResourceType`
- `config:GetComplianceSummaryByConfigRule`
- `config:PutConfigurationRecorder`

Condition context keys for AWS Config

AWS Config has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Cost and Usage Report

AWS Cost and Usage Report (service prefix: cur) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Cost and Usage Report

For more information about controlling access to Customer Usage Reports, see the [Billing and Cost Management Permissions Reference](#) in the *AWS Billing and Cost Management API Reference*.

- `cur>DeleteReportDefinition`
- `cur:DescribeReportDefinitions`
- `cur:PutReportDefinition`

Condition context keys for AWS Cost and Usage Report

AWS Cost and Usage Report has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Cost Explorer Service

AWS Cost Explorer Service (service prefix: ce) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Cost Explorer Service

- `ce:GetTags`
- `ce:GetDimensionValues`
- `ce:GetReservationUtilization`
- `ce:GetCostAndUsage`

Condition context keys for AWS Cost Explorer Service

AWS Cost Explorer Service has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Data Pipeline

Data Pipeline (service prefix: `datipeline`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Data Pipeline

For information about using the following AWS Data Pipeline API actions in an IAM policy, see [Controlling Access to Pipelines and Resources](#) in the *AWS Data Pipeline Developer Guide*.

- `datipeline:QueryObjects`
- `datipeline:GetAccountLimits`
- `datipeline:ValidatePipelineDefinition`
- `datipeline:DeactivatePipeline`
- `datipeline>ListPipelines`
- `datipeline>CreatePipeline`
- `datipeline:ReportTaskProgress`
- `datipeline:ActivatePipeline`
- `datipeline:EvaluateExpression`
- `datipeline:DescribeObjects`
- `datipeline:RemoveTags`
- `datipeline:PutPipelineDefinition`
- `datipeline:GetPipelineDefinition`
- `datipeline:PutAccountLimits`
- `datipeline>DeletePipeline`
- `datipeline:PollForTask`
- `datipeline:SetTaskStatus`
- `datipeline:DescribePipelines`
- `datipeline:SetStatus`
- `datipeline:AddTags`
- `datipeline:ReportTaskRunnerHeartbeat`

Condition context keys for Data Pipeline

Data Pipeline has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- [datipeline:PipelineCreator](#)
- [datipeline:Tag](#)
- [datipeline:workerGroup](#)

Actions and Condition Context Keys for AWS Database Migration Service

AWS Database Migration Service (service prefix: dms) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Database Migration Service

For information about using the following AWS Database Migration Service API actions in an IAM policy, see [IAM Permissions Needed to Use AWS DMS](#) in the *AWS DMS User Guide*.

- [dms:ModifyReplicationTask](#)
- [dms:DescribeConnections](#)
- [dms:DescribeOrderableReplicationInstances](#)
- [dms:DescribeCertificates](#)
- [dms:DescribeEventCategories](#)
- [dms:TestConnection](#)
- [dms:DescribeSchemas](#)
- [dms>CreateReplicationSubnetGroup](#)
- [dms>DeleteReplicationTask](#)
- [dms:DescribeReplicationInstances](#)
- [dms:DescribeReplicationSubnetGroups](#)
- [dms:CreateReplicationTask](#)
- [dms:DescribeEndpoints](#)
- [dms:DescribeReplicationTasks](#)
- [dms:DeleteReplicationInstance](#)
- [dms:CreateEndpoint](#)
- [dms:CreateReplicationInstance](#)
- [dms:ModifyEventSubscription](#)
- [dms:DescribeEventSubscriptions](#)
- [dms:DescribeAccountAttributes](#)
- [dms:DeleteReplicationSubnetGroup](#)
- [dms:DescribeRefreshSchemasStatus](#)
- [dms:DescribeTableStatistics](#)
- [dms:RemoveTagsFromResource](#)
- [dms:DeleteEndpoint](#)
- [dms:DescribeEvents](#)
- [dms:ModifyReplicationInstance](#)
- [dms:StartReplicationTask](#)
- [dms:StopReplicationTask](#)
- [dms:DescribeEndpointTypes](#)
- [dms:DeleteEventSubscription](#)
- [dms:ModifyReplicationSubnetGroup](#)

- [dms:ModifyEndpoint](#)
- [dms:RefreshSchemas](#)
- [dms:AddTagsToResource](#)
- [dms>ListTagsForResource](#)

Condition context keys for AWS Database Migration Service

AWS Database Migration Service has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Device Farm

AWS Device Farm (service prefix: devicefarm) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Device Farm

For more information about the Device Farm API actions in the following list, see [User Access Permissions for AWS Device Farm](#) in the *Device Farm Developer Guide*.

- [devicefarm>DeleteUpload](#)
- [devicefarm>ListSamples](#)
- [devicefarm>ListSuites](#)
- [devicefarm>ListDevices](#)
- [devicefarm>GetDevicePool](#)
- [devicefarm>ListProjects](#)
- [devicefarm>ListRemoteAccessSessions](#)
- [devicefarm>GetJob](#)
- [devicefarm>CreateProject](#)
- [devicefarm>CreateNetworkProfile](#)
- [devicefarm>ListDevicePools](#)
- [devicefarm>DeleteRun](#)
- [devicefarm>StopRemoteAccessSession](#)
- [devicefarm>UpdateDevicePool](#)
- [devicefarm>DeleteDevicePool](#)
- [devicefarm>GetTest](#)
- [devicefarm>GetDevice](#)
- [devicefarm>GetRun](#)
- [devicefarm>ListTests](#)
- [devicefarm>InstallToRemoteAccessSession](#)
- [devicefarm>DeleteRemoteAccessSession](#)
- [devicefarm>GetUpload](#)
- [devicefarm>CreateRemoteAccessSession](#)
- [devicefarm>GetRemoteAccessSession](#)
- [devicefarm>GetSuite](#)
- [devicefarm>ListOfferings](#)
- [devicefarm>ListUniqueProblems](#)
- [devicefarm>ListRuns](#)

- `devicefarm:GetAccountSettings`
- `devicefarm:GetDevicePoolCompatibility`
- `devicefarm:GetProject`
- `devicefarm:StopRun`
- `devicefarm>CreateDevicePool`
- `devicefarm>ListNetworkProfiles`
- `devicefarm>PurchaseOffering`
- `devicefarm:GetNetworkProfile`
- `devicefarm:ScheduleRun`
- `devicefarm:RenewOffering`
- `devicefarm>DeleteProject`
- `devicefarm>ListArtifacts`
- `devicefarm>ListUploads`
- `devicefarm:UpdateProject`
- `devicefarm:GetOfferingStatus`
- `devicefarm:UpdateNetworkProfile`
- `devicefarm>CreateUpload`
- `devicefarm>ListOfferingTransactions`
- `devicefarm>DeleteNetworkProfile`
- `devicefarm>ListJobs`

Condition context keys for AWS Device Farm

AWS Device Farm has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Direct Connect

AWS Direct Connect (service prefix: `directconnect`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Direct Connect

For information about using the following AWS Direct Connect API actions in an IAM policy, see [AWS Direct Connect Actions](#) in the *AWS Direct Connect User Guide*.

- `directconnect>DeleteVirtualInterface`
- `directconnect>DescribeLocations`
- `directconnect>ConfirmPrivateVirtualInterface`
- `directconnect>DescribeConnections`
- `directconnect>CreateConnection`
- `directconnect>DescribeVirtualGateways`
- `directconnect>DeleteInterconnect`
- `directconnect>CreateInterconnect`
- `directconnect>DeleteConnection`
- `directconnect>DescribeConnectionsOnInterconnect`
- `directconnect>ConfirmConnection`
- `directconnect>CreatePrivateVirtualInterface`

- [directconnect:ConfirmPublicVirtualInterface](#)
- [directconnect:DescribeInterconnectLoa](#)
- [directconnect>CreatePublicVirtualInterface](#)
- [directconnect:DescribeConnectionLoa](#)
- [directconnect:DescribeVirtualInterfaces](#)
- [directconnect:DescribeInterconnects](#)
- [directconnect:AllocatePrivateVirtualInterface](#)
- [directconnect:AllocatePublicVirtualInterface](#)
- [directconnect:AllocateConnectionOnInterconnect](#)

Condition context keys for AWS Direct Connect

For information about using conditions in an IAM policy to control access to AWS Direct Connect, see [AWS Direct Connect Keys](#) in the *AWS Direct Connect User Guide*.

AWS Direct Connect has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Directory Service

AWS Directory Service (service prefix: ds) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Directory Service

For information about using the following AWS Directory Service API actions in an IAM policy, see [Controlling Access to AWS Directory Service Resources](#) in the *AWS Directory Service Administration Guide*.

- [ds:RestoreFromSnapshot](#)
- [ds:CancelSchemaExtension](#)
- [ds:GetSnapshotLimits](#)
- [ds>DeleteConditionalForwarder](#)
- [ds:DescribeConditionalForwarders](#)
- [ds:RemoveIpRoutes](#)
- [ds>CreateAlias](#)
- [ds:ConnectDirectory](#)
- [ds:DescribeSnapshots](#)
- [ds>DeleteSnapshot](#)
- [ds:DisableRadius](#)
- [ds:UnauthorizeApplication](#)
- [ds:DescribeDirectories](#)
- [ds>ListSchemaExtensions](#)
- [ds:VerifyTrust](#)
- [ds:AuthorizeApplication](#)
- [ds>ListIpRoutes](#)
- [ds:DescribeTrusts](#)
- [ds>CreateComputer](#)
- [ds:DescribeEventTopics](#)
- [ds>AddIpRoutes](#)

- `ds:EnableRadius`
- `ds:UpdateRadius`
- `ds:RegisterEventTopic`
- `ds:GetDirectoryLimits`
- `ds:EnableSso`
- `ds>CreateMicrosoftAD`
- `ds:RemoveTagsFromResource`
- `ds>CreateDirectory`
- `ds>DeleteDirectory`
- `ds:UpdateConditionalForwarder`
- `ds:StartSchemaExtension`
- `ds:DisableSso`
- `ds>CreateConditionalForwarder`
- `ds:DeregisterEventTopic`
- `ds>ListAuthorizedApplications`
- `ds:AddTagsToResource`
- `ds:DeleteTrust`
- `ds>CreateSnapshot`
- `ds:CreateTrust`
- `ds>ListTagsForResource`

Condition context keys for AWS Directory Service

AWS Directory Service has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon DynamoDB

Amazon DynamoDB (service prefix: dynamodb) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon DynamoDB

For information about using the following DynamoDB API actions in an IAM policy, see [DynamoDB Actions](#) in the *Amazon DynamoDB Developer Guide*.

- `dynamodb:DescribeBackup`
- `dynamodb>ListStreams`
- `dynamodb>ListBackups`
- `dynamodb:Query`
- `dynamodb>DeleteItem`
- `dynamodb>DeleteBackup`
- `dynamodb:TagResource`
- `dynamodb:DescribeTable`
- `dynamodb>CreateTable`
- `dynamodb:GetRecords`
- `dynamodb:GetShardIterator`
- `dynamodb:GetItem`

- `dynamodb>ListTagsOfResource`
- `dynamodb>DescribeContinuousBackups`
- `dynamodb>DescribeReservedCapacity`
- `dynamodb>PurchaseReservedCapacityOfferings`
- `dynamodb>DescribeTimeToLive`
- `dynamodb>BatchGetItem`
- `dynamodb>DescribeStream`
- `dynamodb>BatchWriteItem`
- `dynamodb>DeleteTable`
- `dynamodb>RestoreTableFromBackup`
- `dynamodb>DescribeLimits`
- `dynamodb>UpdateTable`
- `dynamodb>UpdateItem`
- `dynamodb>DescribeReservedCapacityOfferings`
- `dynamodb>ListTables`
- `dynamodb>UntagResource`
- `dynamodb>Scan`
- `dynamodb>PutItem`
- `dynamodb>CreateBackup`

Condition context keys for Amazon DynamoDB

For information about using the following DynamoDB condition keys in an IAM policy, see [IAM Policy Keys](#) in the *Amazon DynamoDB Developer Guide*.

Amazon DynamoDB has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `dynamodb:Attributes`
- `dynamodb:LeadingKeys`
- `dynamodb:ReturnConsumedCapacity`
- `dynamodb:ReturnValues`
- `dynamodb>Select`

Actions and Condition Context Keys for Amazon DynamoDB Accelerator (DAX)

Amazon DynamoDB Accelerator (DAX) (service prefix: `dax`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon DynamoDB Accelerator (DAX)

For information about using the following DynamoDB Accelerator API actions in an IAM policy, see [DynamoDB Actions](#) in the *Amazon DynamoDB Developer Guide*.

- `dax:UpdateSubnetGroup`
- `dax>CreateParameterGroup`
- `dax>DeleteItem`

- `dax:Query`
- `dax:TagResource`
- `dax:DescribeTable`
- `dax:DeleteCluster`
- `dax:GetItem`
- `dax:DescribeParameterGroups`
- `dax:UpdateParameterGroup`
- `dax:DeleteSubnetGroup`
- `dax:DescribeDefaultParameters`
- `dax:IncreaseReplicationFactor`
- `dax:RebootNode`
- `dax:DescribeClusters`
- `dax:BatchGetItem`
- `dax:BatchWriteItem`
- `dax:DeleteParameterGroup`
- `dax>CreateSubnetGroup`
- `dax:DescribeEvents`
- `dax:DecreaseReplicationFactor`
- `dax:UpdateItem`
- `dax>ListTables`
- `dax:UntagResource`
- `dax:DescribeSubnetGroups`
- `dax>ListTags`
- `dax:DescribeParameters`
- `dax:Scan`
- `dax:PutItem`
- `dax:UpdateCluster`
- `dax:CreateCluster`

Condition context keys for Amazon DynamoDB Accelerator (DAX)

Amazon DynamoDB Accelerator (DAX) has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon EC2

Amazon EC2 (service prefix: `ec2`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon EC2

For information about using the following Amazon EC2 API actions in an IAM policy, see [Actions for Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*.

- `ec2:ModifySnapshotAttribute`
- `ec2:DescribeSpotDatafeedSubscription`
- `ec2:ModifyHosts`
- `ec2:DescribeSnapshots`

- `ec2:CreateNatGateway`
- `ec2:DescribeVpcEndpointConnections`
- `ec2:ResetInstanceAttribute`
- `ec2:ResetSnapshotAttribute`
- `ec2:PurchaseReservedInstancesOffering`
- `ec2:RevokeSecurityGroupEgress`
- `ec2:DescribeSpotPriceHistory`
- `ec2:DescribeHosts`
- `ec2:CreateVpcPeeringConnection`
- `ec2:CreateDefaultVpc`
- `ec2:PurchaseScheduledInstances`
- `ec2:CopyFpgaImage`
- `ec2:DescribeIdentityIdFormat`
- `ec2:DeleteFlowLogs`
- `ec2:CreateDefaultSubnet`
- `ec2:DescribeInstances`
- `ec2:CreateReservedInstancesListing`
- `ec2:DescribeSecurityGroupReferences`
- `ec2:DescribeDhcpOptions`
- `ec2:AttachVolume`
- `ec2:CreateVpcEndpointServiceConfiguration`
- `ec2:ModifyReservedInstances`
- `ec2:AllocateHosts`
- `ec2:DeleteNetworkInterface`
- `ec2:GetConsoleScreenshot`
- `ec2:AssignPrivateIpAddresses`
- `ec2:DeleteSnapshot`
- `ec2:ModifyVpcPeeringConnectionOptions`
- `ec2:DeleteSecurityGroup`
- `ec2:ModifyVpcEndpointServicePermissions`
- `ec2:AllocateAddress`
- `ec2:ResetFpgaImageAttribute`
- `ec2:DeleteFpgaImage`
- `ec2:DescribeVolumeAttribute`
- `ec2:ConfirmProductInstance`
- `ec2:DeleteInternetGateway`
- `ec2:DescribeImportSnapshotTasks`
- `ec2:RevokeSecurityGroupIngress`
- `ec2:ReportInstanceStatus`
- `ec2:DescribeInstanceCreditSpecifications`
- `ec2:ModifyInstanceAttribute`
- `ec2:EnableVpcClassicLink`
- `ec2:DeleteVpnConnectionRoute`
- `ec2:DescribeSecurityGroups`
- `ec2:ModifySubnetAttribute`

- `ec2:CancelSpotFleetRequests`
- `ec2:GetPasswordData`
- `ec2:DescribeIdFormat`
- `ec2:DisableVpcClassicLink`
- `ec2:ModifyVolumeAttribute`
- `ec2:DescribeSubnets`
- `ec2>CreateEgressOnlyInternetGateway`
- `ec2:DescribeFlowLogs`
- `ec2:CreateDhcpOptions`
- `ec2:DescribeFpgaImages`
- `ec2:RunInstances`
- `ec2:DeleteVpcEndpointConnectionNotifications`
- `ec2:CreateLaunchTemplateVersion`
- `ec2:DescribeVpcPeeringConnections`
- `ec2:DescribeVpcEndpointServicePermissions`
- `ec2:CancelSpotInstanceRequests`
- `ec2:DescribeHostReservationOfferings`
- `ec2:ModifyVpcEndpointConnectionNotification`
- `ec2:AssociateDhcpOptions`
- `ec2:ReplaceNetworkAclEntry`
- `ec2:DescribeVpcEndpoints`
- `ec2:DescribeStaleSecurityGroups`
- `ec2:CreateVpnConnectionRoute`
- `ec2:DescribeVpcClassicLinkDnsSupport`
- `ec2:UpdateSecurityGroupRuleDescriptionsEgress`
- `ec2:ModifyInstanceCreditSpecification`
- `ec2:RunScheduledInstances`
- `ec2:ReplaceRoute`
- `ec2:DeleteNatGateway`
- `ec2:DescribePlacementGroups`
- `ec2:CreateSpotDatafeedSubscription`
- `ec2:DescribeReservedInstancesOfferings`
- `ec2:DetachVpnGateway`
- `ec2:CancelBundleTask`
- `ec2:DisableVpcClassicLinkDnsSupport`
- `ec2:DescribeSpotFleetInstances`
- `ec2:DescribeFpgaImageAttribute`
- `ec2:UpdateSecurityGroupRuleDescriptionsIngress`
- `ec2:ReplaceIamInstanceProfileAssociation`
- `ec2:DisassociateIamInstanceProfile`
- `ec2:RebootInstances`
- `ec2:DetachClassicLinkVpc`
- `ec2:ModifyFpgaImageAttribute`
- `ec2:DescribeVpcs`
- `ec2:GetReservedInstancesExchangeQuote`

- [ec2:MoveAddressToVpc](#)
- [ec2:DescribeAddresses](#)
- [ec2:StartInstances](#)
- [ec2>CreateSubnet](#)
- [ec2:DisassociateAddress](#)
- [ec2>DeleteLaunchTemplateVersions](#)
- [ec2:RestoreAddressToClassic](#)
- [ec2:AcceptVpcEndpointConnections](#)
- [ec2:DescribeScheduledInstances](#)
- [ec2:DescribeIamInstanceProfileAssociations](#)
- [ec2:ImportInstance](#)
- [ec2:DescribeElasticGpus](#)
- [ec2:DeleteRoute](#)
- [ec2:BundleInstance](#)
- [ec2:DeletePlacementGroup](#)
- [ec2:DisassociateVpcCidrBlock](#)
- [ec2:DisableVgwRoutePropagation](#)
- [ec2:CreateNetworkAclEntry](#)
- [ec2:AuthorizeSecurityGroupIngress](#)
- [ec2:DeleteCustomerGateway](#)
- [ec2:CreateFlowLogs](#)
- [ec2:DescribeNetworkInterfaceAttribute](#)
- [ec2:DescribeConversionTasks](#)
- [ec2:DescribeCustomerGateways](#)
- [ec2:ModifySpotFleetRequest](#)
- [ec2:DescribeSpotFleetRequests](#)
- [ec2:DescribeNetworkInterfacePermissions](#)
- [ec2:DescribeVolumesModifications](#)
- [ec2:DetachInternetGateway](#)
- [ec2:DeleteVpc](#)
- [ec2:StopInstances](#)
- [ec2:RequestSpotFleet](#)
- [ec2:DescribeExportTasks](#)
- [ec2:CreateRouteTable](#)
- [ec2:AssociateAddress](#)
- [ec2:CreateCustomerGateway](#)
- [ec2:DescribeReservedInstancesListings](#)
- [ec2:DescribeTags](#)
- [ec2:DeleteEgressOnlyInternetGateway](#)
- [ec2:DeleteDhcpOptions](#)
- [ec2:DescribeImages](#)
- [ec2:AuthorizeSecurityGroupEgress](#)
- [ec2:ImportImage](#)
- [ec2:RejectVpcEndpointConnections](#)
- [ec2:DeleteVpcPeeringConnection](#)

- `ec2:ModifyVpcEndpointServiceConfiguration`
- `ec2:ReleaseAddress`
- `ec2:ResetNetworkInterfaceAttribute`
- `ec2>CreateVpc`
- `ec2:CopyImage`
- `ec2:DescribeHostReservations`
- `ec2:DeleteNetworkAcl`
- `ec2:CreateTags`
- `ec2:CreateSnapshot`
- `ec2:ImportVolume`
- `ec2:DeleteKeyPair`
- `ec2:DescribeImageAttribute`
- `ec2:ModifyVpcTenancy`
- `ec2:DeleteVpnGateway`
- `ec2:UnassignPrivateIpAddresses`
- `ec2:DescribeImportImageTasks`
- `ec2:AssociateVpcCidrBlock`
- `ec2:ImportKeyPair`
- `ec2:DescribeVpnConnections`
- `ec2:AcceptVpcPeeringConnection`
- `ec2:DeleteLaunchTemplate`
- `ec2:ModifyVpcEndpoint`
- `ec2:CreateImage`
- `ec2:CancelImportTask`
- `ec2:CancelConversionTask`
- `ec2:DescribeInstanceAttribute`
- `ec2:DescribeSpotFleetRequestHistory`
- `ec2:ModifyImageAttribute`
- `ec2:DescribeInstanceStatus`
- `ec2:DescribeNatGateways`
- `ec2:CreateVpcEndpoint`
- `ec2:ModifyVolume`
- `ec2:ResetImageAttribute`
- `ec2:AssociateIamInstanceProfile`
- `ec2:DescribeVpcAttribute`
- `ec2:RejectVpcPeeringConnection`
- `ec2:CreateVpnConnection`
- `ec2:DescribeVolumes`
- `ec2:DeleteVpnConnection`
- `ec2:AttachVpnGateway`
- `ec2:CreateSecurityGroup`
- `ec2:ReleaseHosts`
- `ec2:DeleteTags`
- `ec2:CreateVolume`
- `ec2:ModifyLaunchTemplate`

- [ec2:ReplaceRouteTableAssociation](#)
- [ec2:PurchaseHostReservation](#)
- [ec2:ModifyVpcAttribute](#)
- [ec2>CreateNetworkInterfacePermission](#)
- [ec2>DeleteSpotDatafeedSubscription](#)
- [ec2>CreateVpcEndpointConnectionNotification](#)
- [ec2>DeleteSubnet](#)
- [ec2:CopySnapshot](#)
- [ec2:DescribeVpcEndpointServiceConfigurations](#)
- [ec2:DeleteVolume](#)
- [ec2:DescribeReservedInstances](#)
- [ec2:DescribeVpcEndpointServices](#)
- [ec2:TerminateInstances](#)
- [ec2:DescribeAvailabilityZones](#)
- [ec2:DescribeNetworkAcls](#)
- [ec2:DescribeAccountAttributes](#)
- [ec2:CreateNetworkAcl](#)
- [ec2:EnableVgwRoutePropagation](#)
- [ec2:DescribeVpcEndpointConnectionNotifications](#)
- [ec2:GetConsoleOutput](#)
- [ec2:DescribeRouteTables](#)
- [ec2:CancelExportTask](#)
- [ec2:DescribeKeyPairs](#)
- [ec2:AcceptReservedInstancesExchangeQuote](#)
- [ec2:UnmonitorInstances](#)
- [ec2:AssociateRouteTable](#)
- [ec2:DeleteNetworkAclEntry](#)
- [ec2:CreateFpgaImage](#)
- [ec2:AssignIpv6Addresses](#)
- [ec2:DeregisterImage](#)
- [ec2:CreatePlacementGroup](#)
- [ec2:UnassignIpv6Addresses](#)
- [ec2:DeleteVpcEndpoints](#)
- [ec2:DescribeSpotInstanceRequests](#)
- [ec2:DeleteVpcEndpointServiceConfigurations](#)
- [ec2:AttachNetworkInterface](#)
- [ec2:DescribeInternetGateways](#)
- [ec2:CreateRoute](#)
- [ec2:ModifyIdentityIdFormat](#)
- [ec2:DescribeVolumeStatus](#)
- [ec2:ModifyIdFormat](#)
- [ec2:CreateInstanceExportTask](#)
- [ec2:EnableVolumeIO](#)
- [ec2:DeleteRouteTable](#)
- [ec2:AttachInternetGateway](#)

- `ec2:DescribePrefixLists`
- `ec2:GetHostReservationPurchasePreview`
- `ec2:DescribeReservedInstancesModifications`
- `ec2>CreateInternetGateway`
- `ec2:RegisterImage`
- `ec2:ModifyInstancePlacement`
- `ec2:DescribeRegions`
- `ec2:AssociateSubnetCidrBlock`
- `ec2:DescribeSnapshotAttribute`
- `ec2:DescribeMovingAddresses`
- `ec2:DescribeClassicLinkInstances`
- `ec2:DescribeBundleTasks`
- `ec2:ImportSnapshot`
- `ec2:DescribeScheduledInstanceAvailability`
- `ec2:DetachVolume`
- `ec2:DescribeVpnGateways`
- `ec2>CreateNetworkInterface`
- `ec2>DeleteNetworkInterfacePermission`
- `ec2:EnableVpcClassicLinkDnsSupport`
- `ec2:ReplaceNetworkAclAssociation`
- `ec2:RequestSpotInstances`
- `ec2:DescribeVpcClassicLink`
- `ec2:DescribeNetworkInterfaces`
- `ec2:ModifyNetworkInterfaceAttribute`
- `ec2:DetachNetworkInterface`
- `ec2:DisassociateRouteTable`
- `ec2:CreateKeyPair`
- `ec2:AttachClassicLinkVpc`
- `ec2:DescribeEgressOnlyInternetGateways`
- `ec2:DisassociateSubnetCidrBlock`
- `ec2:CancelReservedInstancesListing`
- `ec2:MonitorInstances`
- `ec2:CreateVpnGateway`

Condition context keys for Amazon EC2

For information about using the following Amazon EC2 conditions in an IAM policy, see [Condition Keys for Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*.

Amazon EC2 has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `aws:RequestTag/tag-key`
- `aws:TagKeys`
- `ec2:AccepterVpc`
- `ec2:AuthorizedService`
- `ec2:AuthorizedUser`

- `ec2:AvailabilityZone`
- `ec2:CreateAction`
- `ec2:EbsOptimized`
- `ec2:Encrypted`
- `ec2:ImageType`
- `ec2:InstanceProfile`
- `ec2:InstanceType`
- `ec2:Owner`
- `ec2:ParentSnapshot`
- `ec2:ParentVolume`
- `ec2:Permission`
- `ec2:PlacementGroup`
- `ec2:PlacementGroupStrategy`
- `ec2:Public`
- `ec2:Region`
- `ec2:RequesterVpc`
- `ec2:ReservedInstancesOfferingType`
- `ec2:ResourceTag/`
- `ec2:ResourceTag/tag-key`
- `ec2:RootDeviceType`
- `ec2:SnapshotTime`
- `ec2:Subnet`
- `ec2:Tenancy`
- `ec2:VolumeIops`
- `ec2:VolumeSize`
- `ec2:VolumeType`
- `ec2:Vpc`

Actions and Condition Context Keys for Amazon EC2 Container Registry

Amazon EC2 Container Registry (service prefix: `ecr`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon EC2 Container Registry

For information about controlling access to Amazon ECR by using an IAM policy, see [Amazon ECR IAM Policies and Roles](#) in the *Amazon Elastic Container Registry User Guide*.

- `ecr:UploadLayerPart`
- `ecr:BatchGetImage`
- `ecr:SetRepositoryPolicy`
- `ecr:CompleteLayerUpload`
- `ecr:DescribeImages`
- `ecr:BatchCheckLayerAvailability`
- `ecr:GetDownloadUrlForLayer`
- `ecr:InitiateLayerUpload`

- `ecr:PutImage`
- `ecr>ListImages`
- `ecr:GetRepositoryPolicy`
- `ecr>DeleteRepositoryPolicy`
- `ecr:DescribeRepositories`
- `ecr:BatchDeleteImage`
- `ecr:DeleteRepository`
- `ecr:GetAuthorizationToken`
- `ecr>CreateRepository`

Condition context keys for Amazon EC2 Container Registry

Amazon EC2 Container Registry has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon EC2 Container Service

Amazon EC2 Container Service (service prefix: `ecs`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon EC2 Container Service

For information about using the following Amazon ECS API actions in an IAM policy, see [Amazon ECS IAM Policies](#) in the *Amazon ECS Developer Guide*.

- `ecs:RegisterTaskDefinition`
- `ecs:DescribeContainerInstances`
- `ecs>DeleteService`
- `ecs:DescribeTasks`
- `ecs:DeregisterContainerInstance`
- `ecs:SubmitTaskStateChange`
- `ecs>DeleteCluster`
- `ecs:RunTask`
- `ecs:StartTelemetrySession`
- `ecs:DeregisterTaskDefinition`
- `ecs:StopTask`
- `ecs>ListClusters`
- `ecs:DescribeServices`
- `ecs:DescribeClusters`
- `ecs:StartTask`
- `ecs>ListTaskDefinitions`
- `ecs:DiscoverPollEndpoint`
- `ecs:UpdateContainerInstancesState`
- `ecs>ListContainerInstances`
- `ecs>ListServices`
- `ecs:SubmitContainerStateChange`
- `ecs>ListTasks`

- `ecs:DescribeTaskDefinition`
- `ecs:UpdateContainerAgent`
- `ecs>CreateService`
- `ecs:UpdateService`
- `ecs:RegisterContainerInstance`
- `ecs:Poll`
- `ecs>ListTaskDefinitionFamilies`
- `ecs>CreateCluster`

Condition context keys for Amazon EC2 Container Service

For information about using the following Amazon ECS conditions in an IAM policy, see [Amazon ECS IAM Policies](#) in the *Amazon ECS Developer Guide*.

Amazon EC2 Container Service has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `ecs:cluster`
- `ecs:container-instances`

Actions and Condition Context Keys for AWS Elastic Beanstalk

AWS Elastic Beanstalk (service prefix: elasticbeanstalk) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Elastic Beanstalk

For information about using the following Elastic Beanstalk API actions in an IAM policy, see [Policy Information for Elastic Beanstalk Actions](#) in the *AWS Elastic Beanstalk Developer Guide*.

- `elasticbeanstalk:DescribeEnvironments`
- `elasticbeanstalk:DescribeConfigurationSettings`
- `elasticbeanstalk>DeleteEnvironmentConfiguration`
- `elasticbeanstalk>ListPlatformVersions`
- `elasticbeanstalk:SwapEnvironmentCNAMEs`
- `elasticbeanstalk:DescribeInstancesHealth`
- `elasticbeanstalk>CreatePlatformVersion`
- `elasticbeanstalk>DeleteApplicationVersion`
- `elasticbeanstalk>CreateConfigurationTemplate`
- `elasticbeanstalk:DescribeApplications`
- `elasticbeanstalk>ListAvailableSolutionStacks`
- `elasticbeanstalk:UpdateConfigurationTemplate`
- `elasticbeanstalk:TerminateEnvironment`
- `elasticbeanstalk:ComposeEnvironments`
- `elasticbeanstalk:RebuildEnvironment`
- `elasticbeanstalk>CreateApplication`
- `elasticbeanstalk>CreateEnvironment`
- `elasticbeanstalk:UpdateApplicationResourceLifecycle`
- `elasticbeanstalk:AbortEnvironmentUpdate`

- elasticbeanstalk:RequestEnvironmentInfo
- elasticbeanstalk:DescribeConfigurationOptions
- elasticbeanstalk:DescribeEnvironmentManagedActionHistory
- elasticbeanstalk:UpdateEnvironment
- elasticbeanstalk:CreateStorageLocation
- elasticbeanstalk:UpdateApplication
- elasticbeanstalk:ApplyEnvironmentManagedAction
- elasticbeanstalk:DescribeEnvironmentManagedActions
- elasticbeanstalk:CheckDNSAvailability
- elasticbeanstalk:RestartAppServer
- elasticbeanstalk:CreateApplicationVersion
- elasticbeanstalk:RetrieveEnvironmentInfo
- elasticbeanstalk:DescribeEnvironmentResources
- elasticbeanstalk:DeletePlatformVersion
- elasticbeanstalk:DescribeEvents
- elasticbeanstalk:ValidateConfigurationSettings
- elasticbeanstalk:UpdateApplicationVersion
- elasticbeanstalk:DescribePlatformVersion
- elasticbeanstalk:DescribeApplicationVersions
- elasticbeanstalk:DescribeEnvironmentHealth
- elasticbeanstalk:DeleteApplication
- elasticbeanstalk:DeleteConfigurationTemplate

Condition context keys for AWS Elastic Beanstalk

For information about using the following Elastic Beanstalk conditions in an IAM policy, see [Condition Keys for Elastic Beanstalk Actions](#) in the *AWS Elastic Beanstalk Developer Guide*.

AWS Elastic Beanstalk has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- elasticbeanstalk:FromApplication
- elasticbeanstalk:FromApplicationVersion
- elasticbeanstalk:FromConfigurationTemplate
- elasticbeanstalk:FromEnvironment
- elasticbeanstalk:FromSolutionStack
- elasticbeanstalk:InApplication

Actions and Condition Context Keys for Amazon Elastic File System

Amazon Elastic File System (service prefix: elasticfilesystem) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Elastic File System

For information about using the following Amazon EFS API actions in an IAM policy, see [Authentication and Access Control](#) in the *Amazon Elastic File System User Guide*.

- `elasticfilesystem:ModifyMountTargetSecurityGroups`
- `elasticfilesystem:DescribeTags`
- `elasticfilesystem:DescribeMountTargetSecurityGroups`
- `elasticfilesystem:DescribeFileSystems`
- `elasticfilesystem>CreateMountTarget`
- `elasticfilesystem:DescribeMountTargets`
- `elasticfilesystem:CreateFileSystem`
- `elasticfilesystem:DeleteTags`
- `elasticfilesystem:CreateTags`
- `elasticfilesystem:DeleteMountTarget`
- `elasticfilesystem:DeleteFileSystem`

Condition context keys for Amazon Elastic File System

Amazon Elastic File System has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Elastic Load Balancing

Elastic Load Balancing (service prefix: `elasticloadbalancing`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Elastic Load Balancing

For information about using the following Elastic Load Balancing API actions in an IAM policy, see [Specifying ELB Actions in an IAM Policy](#) in the *Elastic Load Balancing User Guide*.

- `elasticloadbalancing:AddTags`
- `elasticloadbalancing:AddTags`
- `elasticloadbalancing:ApplySecurityGroupsToLoadBalancer`
- `elasticloadbalancing:AttachLoadBalancerToSubnets`
- `elasticloadbalancing:ConfigureHealthCheck`
- `elasticloadbalancing>CreateAppCookieStickinessPolicy`
- `elasticloadbalancing>CreateLBCookieStickinessPolicy`
- `elasticloadbalancing>CreateListener`
- `elasticloadbalancing>CreateLoadBalancer`
- `elasticloadbalancing>CreateLoadBalancer`
- `elasticloadbalancing>CreateLoadBalancerListeners`
- `elasticloadbalancing>CreateLoadBalancerPolicy`
- `elasticloadbalancing>CreateRule`
- `elasticloadbalancing>CreateTargetGroup`
- `elasticloadbalancing>DeleteListener`
- `elasticloadbalancing>DeleteLoadBalancer`
- `elasticloadbalancing>DeleteLoadBalancer`
- `elasticloadbalancing>DeleteLoadBalancerListeners`
- `elasticloadbalancing>DeleteLoadBalancerPolicy`
- `elasticloadbalancing>DeleteRule`
- `elasticloadbalancing>DeleteTargetGroup`

- `elasticloadbalancing:DeregisterInstancesFromLoadBalancer`
- `elasticloadbalancing:DeregisterTargets`
- `elasticloadbalancing:DescribeInstanceHealth`
- `elasticloadbalancing:DescribeListeners`
- `elasticloadbalancing:DescribeLoadBalancerAttributes`
- `elasticloadbalancing:DescribeLoadBalancerAttributes`
- `elasticloadbalancing:DescribeLoadBalancerPolicies`
- `elasticloadbalancing:DescribeLoadBalancerPolicyTypes`
- `elasticloadbalancing:DescribeLoadBalancers`
- `elasticloadbalancing:DescribeLoadBalancers`
- `elasticloadbalancing:DescribeRules`
- `elasticloadbalancing:DescribeSSLPolicies`
- `elasticloadbalancing:DescribeTags`
- `elasticloadbalancing:DescribeTags`
- `elasticloadbalancing:DescribeTargetGroupAttributes`
- `elasticloadbalancing:DescribeTargetGroups`
- `elasticloadbalancing:DescribeTargetHealth`
- `elasticloadbalancing:DetachLoadBalancerFromSubnets`
- `elasticloadbalancing:DisableAvailabilityZonesForLoadBalancer`
- `elasticloadbalancing:EnableAvailabilityZonesForLoadBalancer`
- `elasticloadbalancing:ModifyListener`
- `elasticloadbalancing:ModifyLoadBalancerAttributes`
- `elasticloadbalancing:ModifyLoadBalancerAttributes`
- `elasticloadbalancing:ModifyRule`
- `elasticloadbalancing:ModifyTargetGroup`
- `elasticloadbalancing:ModifyTargetGroupAttributes`
- `elasticloadbalancing:RegisterInstancesWithLoadBalancer`
- `elasticloadbalancing:RegisterTargets`
- `elasticloadbalancing:RemoveTags`
- `elasticloadbalancing:RemoveTags`
- `elasticloadbalancing:SetIpAddressType`
- `elasticloadbalancing:SetLoadBalancerListenerSSLCertificate`
- `elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer`
- `elasticloadbalancing:SetLoadBalancerPoliciesOfListener`
- `elasticloadbalancing:SetRulePriorities`
- `elasticloadbalancing:SetSecurityGroups`
- `elasticloadbalancing:SetSubnets`

Condition context keys for Elastic Load Balancing

For information about using conditions in an IAM policy to control access to Elastic Load Balancing, see [Specifying Condition Keys in an IAM Policy](#) in the *Elastic Load Balancing User Guide*.

Elastic Load Balancing has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Elastic MapReduce

Amazon Elastic MapReduce (service prefix: elasticmapreduce) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Elastic MapReduce

For information about controlling access to Amazon EMR by specifying actions in an IAM policy, see [Set Access Policies for IAM Users](#) in the *Amazon EMR Developer Guide*.

- `elasticmapreduce:DescribeCluster`
- `elasticmapreduce:DescribeJobFlows`
- `elasticmapreduce:DeleteSecurityConfiguration`
- `elasticmapreduce:SetTerminationProtection`
- `elasticmapreduce:ViewEventsFromAllClustersInConsole`
- `elasticmapreduce:ModifyInstanceGroups`
- `elasticmapreduce:TerminateJobFlows`
- `elasticmapreduce>ListInstanceGroups`
- `elasticmapreduce>ListBootstrapActions`
- `elasticmapreduce:RemoveTags`
- `elasticmapreduce:DescribeStep`
- `elasticmapreduce>ListInstances`
- `elasticmapreduce>ListSteps`
- `elasticmapreduce>ListClusters`
- `elasticmapreduce:RemoveAutoScalingPolicy`
- `elasticmapreduce>ListSecurityConfigurations`
- `elasticmapreduce:RunJobFlow`
- `elasticmapreduce:DescribeSecurityConfiguration`
- `elasticmapreduce:AddJobFlowSteps`
- `elasticmapreduce>CreateSecurityConfiguration`
- `elasticmapreduce:AddTags`
- `elasticmapreduce>AddInstanceGroups`
- `elasticmapreduce:PutAutoScalingPolicy`
- `elasticmapreduce:SetVisibleToAllUsers`
- `elasticmapreduce:CancelSteps`

Condition context keys for Amazon Elastic MapReduce

Amazon Elastic MapReduce has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Elastic Transcoder

Amazon Elastic Transcoder (service prefix: elastictranscoder) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Elastic Transcoder

For information about using the following Elastic Transcoder API actions in an IAM policy, see [List of Elastic Transcoder Operations with Controllable Access](#) in the *Amazon Elastic Transcoder Developer Guide*.

- `elastictranscoder>ListJobsByStatus`
- `elastictranscoder>ListJobsByPipeline`
- `elastictranscoder>ReadPipeline`
- `elastictranscoder>CreatePipeline`
- `elastictranscoder>ListPipelines`
- `elastictranscoder>CreatePreset`
- `elastictranscoder>UpdatePipelineNotifications`
- `elastictranscoder>ReadPreset`
- `elastictranscoder>DeletePipeline`
- `elastictranscoder>TestRole`
- `elastictranscoder>DeletePreset`
- `elastictranscoder>CancelJob`
- `elastictranscoder>ReadJob`
- `elastictranscoder>ListPresets`
- `elastictranscoder>UpdatePipeline`
- `elastictranscoder>CreateJob`
- `elastictranscoder>UpdatePipelineStatus`

Condition context keys for Amazon Elastic Transcoder

Amazon Elastic Transcoder has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon ElastiCache

Amazon ElastiCache (service prefix: `elasticache`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon ElastiCache

When you create an ElastiCache policy in IAM you must use the "*" wildcard character for the Resource block. For information about using the following ElastiCache API actions in an IAM policy, see [ElastiCache Actions and IAM](#) in the *Amazon ElastiCache User Guide*.

- `elasticache>DeleteCacheParameterGroup`
- `elasticache>ModifyReplicationGroup`
- `elasticache>CreateCacheCluster`
- `elasticache>DescribeReservedCacheNodes`
- `elasticache>DeleteReplicationGroup`
- `elasticache>RevokeCacheSecurityGroupIngress`
- `elasticache>DescribeCacheSecurityGroups`
- `elasticache>ListAllowedNodeTypeModifications`
- `elasticache>DescribeReplicationGroups`
- `elasticache>DescribeCacheEngineVersions`

- `elasticache:DescribeSnapshots`
- `elasticache>DeleteSnapshot`
- `elasticache:DescribeCacheSubnetGroups`
- `elasticache:CopySnapshot`
- `elasticache:DescribeCacheParameters`
- `elasticache>CreateCacheSecurityGroup`
- `elasticache:DeleteCacheSubnetGroup`
- `elasticache:RebootCacheCluster`
- `elasticache:DeleteCacheSecurityGroup`
- `elasticache:DescribeReservedCacheNodesOfferings`
- `elasticache:PurchaseReservedCacheNodesOffering`
- `elasticache:ModifyCacheCluster`
- `elasticache>CreateCacheParameterGroup`
- `elasticache:DescribeCacheParameterGroups`
- `elasticache>CreateCacheSubnetGroup`
- `elasticache:RemoveTagsFromResource`
- `elasticache>CreateReplicationGroup`
- `elasticache:DescribeEvents`
- `elasticache:DescribeEngineDefaultParameters`
- `elasticache:DeleteCacheCluster`
- `elasticache:AuthorizeCacheSecurityGroupIngress`
- `elasticache:DescribeCacheClusters`
- `elasticache:ResetCacheParameterGroup`
- `elasticache:AddTagsToResource`
- `elasticache:CreateSnapshot`
- `elasticache:ModifyCacheParameterGroup`
- `elasticache>ListTagsForResource`
- `elasticache:ModifyCacheSubnetGroup`

Condition context keys for Amazon ElastiCache

For information about conditions in an IAM policy to control access to ElastiCache, see [ElastiCache Keys in the *Amazon ElastiCache User Guide*](#).

Amazon ElastiCache has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Elasticsearch Service

Amazon Elasticsearch Service (service prefix: es) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Elasticsearch Service

For information about using the following Amazon ElasticSearch API actions in an IAM policy, see [Configuring an Access Policy for an Elasticsearch Domain](#) in the *Amazon Elasticsearch Developer Guide*.

- `es:UpdateElasticsearchDomainConfig`
- `es:ESHttpPut`
- `es:DescribeElasticsearchDomain`
- `es:DeleteElasticsearchDomain`
- `es:DescribeElasticsearchDomains`
- `es:RemoveTags`
- `es:ESHttpPost`
- `es>CreateElasticsearchDomain`
- `es:ESHttpHead`
- `es>ListTags`
- `es:AddTags`
- `es>ListDomainNames`
- `es:ESHttpGet`
- `es:DescribeElasticsearchDomainConfig`
- `es:ESHttpDelete`

Condition context keys for Amazon Elasticsearch Service

Amazon Elasticsearch Service has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Elemental MediaConvert

AWS Elemental MediaConvert (service prefix: `mediacconvert`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Elemental MediaConvert

- `mediacconvert:DeleteJobTemplate`
- `mediacconvert:DeleteQueue`
- `mediacconvert:DescribeEndpoint`
- `mediacconvert>ListJobTemplates`
- `mediacconvert:UpdateJobTemplate`
- `mediacconvert:GetJob`
- `mediacconvert:GetPreset`
- `mediacconvert>CreateQueue`
- `mediacconvert>CreatePreset`
- `mediacconvert:GetJobTemplate`
- `mediacconvert>CreateJobTemplate`
- `mediacconvert>ListQueues`
- `mediacconvert>DeletePreset`
- `mediacconvert:UpdatePreset`
- `mediacconvert:CancelJob`
- `mediacconvert:GetQueue`
- `mediacconvert>ListPresets`

- `mediaconvert:CreateJob`
- `mediaconvert>ListJobs`
- `mediaconvert:UpdateQueue`

Condition context keys for AWS Elemental MediaConvert

AWS Elemental MediaConvert has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Elemental MediaLive

AWS Elemental MediaLive (service prefix: `medialive`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Elemental MediaLive

- `medialive:StopChannel`
- `medialive>ListInputSecurityGroups`
- `medialive:DescribeInputSecurityGroup`
- `medialive:DescribeChannel`
- `medialive>ListChannels`
- `medialive>ListInputs`
- `medialive>CreateInput`
- `medialive>DeleteInput`
- `medialive>DeleteChannel`
- `medialive>DeleteInputSecurityGroup`
- `medialive:DescribeInput`
- `medialive:StartChannel`
- `medialive>CreateChannel`
- `medialive>CreateInputSecurityGroup`

Condition context keys for AWS Elemental MediaLive

AWS Elemental MediaLive has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Elemental MediaPackage

AWS Elemental MediaPackage (service prefix: `mediapackage`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Elemental MediaPackage

- `mediapackage:DescribeChannel`
- `mediapackage>DeleteOriginEndpoint`
- `mediapackage>CreateOriginEndpoint`

- mediapackage:DescribeOriginEndpoint
- mediapackage>CreateChannel
- mediapackage>ListChannels
- mediapackage>ListOriginEndpoints
- mediapackage:UpdateOriginEndpoint
- mediapackage:UpdateChannel
- mediapackage>DeleteChannel

Condition context keys for AWS Elemental MediaPackage

AWS Elemental MediaPackage has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Elemental MediaStore

AWS Elemental MediaStore (service prefix: mediastore) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Elemental MediaStore

- mediastore>DeleteObject
- mediastore>ListItems
- mediastore>DeleteContainer
- mediastore>DescribeContainer
- mediastore>PutContainerPolicy
- mediastore>DescribeObject
- mediastore>DeleteContainerPolicy
- mediastore>GetContainerPolicy
- mediastore>PutObject
- mediastore>CreateContainer
- mediastore>GetObject
- mediastore>ListContainers

Condition context keys for AWS Elemental MediaStore

AWS Elemental MediaStore has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon FreeRTOS

Amazon FreeRTOS (service prefix: freertos) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon FreeRTOS

- freertos>DeleteSoftwareConfiguration
- freertos>ListSoftwareConfigurations

- `freertos:GetSoftwareURLForConfiguration`
- `freertos:UpdateSoftwareConfiguration`
- `freertos:DescribeSoftwareConfiguration`
- `freertos>ListHardwareVendors`
- `freertos>ListFreeRTOSVersions`
- `freertos>ListHardwarePlatforms`
- `freertos>CreateSoftwareConfiguration`
- `freertos:GetSoftwareURL`
- `freertos:DescribeHardwarePlatform`

Condition context keys for Amazon FreeRTOS

Amazon FreeRTOS has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon GameLift

Amazon GameLift (service prefix: `gamelift`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon GameLift

For information about using the following Amazon GameLift API actions in an IAM policy, see [Amazon GameLift IAM Policy Examples](#) in the *Amazon GameLift Developer Guide*.

- `gamelift>CreatePlayerSessions`
- `gamelift/SearchGameSessions`
- `gamelift:UpdateFleetCapacity`
- `gamelift>DeleteScalingPolicy`
- `gamelift>CreateAlias`
- `gamelift>CreateFleet`
- `gamelift:DescribeGameSessionDetails`
- `gamelift:DescribeFleetCapacity`
- `gamelift>DeleteBuild`
- `gamelift:RequestUploadCredentials`
- `gamelift:DescribeFleetEvents`
- `gamelift:UpdateRuntimeConfiguration`
- `gamelift>CreateBuild`
- `gamelift:DescribeScalingPolicies`
- `gamelift:DescribeRuntimeConfiguration`
- `gamelift:PutScalingPolicy`
- `gamelift>ListAliases`
- `gamelift:DescribePlayerSessions`
- `gamelift:DescribeFleetPortSettings`
- `gamelift:DescribeFleetUtilization`
- `gamelift>DeleteAlias`
- `gamelift:UpdateBuild`

- gamelift:DescribeFleetAttributes
- gamelift:DescribeEC2InstanceLimits
- gamelift:UpdateGameSession
- gamelift:DeleteFleet
- gamelift>CreatePlayerSession
- gamelift:ResolveAlias
- gamelift>ListFleets
- gamelift:GetInstanceAccess
- gamelift:UpdateFleetPortSettings
- gamelift:DescribeBuild
- gamelift:GetGameSessionLogUrl
- gamelift:UpdateFleetAttributes
- gamelift:CreateGameSession
- gamelift:UpdateAlias
- gamelift:DescribeInstances
- gamelift>ListBuilds
- gamelift:DescribeAlias
- gamelift:DescribeGameSessions

Condition context keys for Amazon GameLift

Amazon GameLift has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Glacier

Amazon Glacier (service prefix: glacier) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Glacier

For information about using the following Amazon Glacier API actions in an IAM policy, see [Access Control Using AWS Identity and Access Management \(IAM\)](#) in the *Amazon Glacier Developer Guide*.

- glacier:PurchaseProvisionedCapacity
- glacier:DescribeJob
- glacier:GetJobOutput
- glacier>CreateVault
- glacier:GetDataRetrievalPolicy
- glacier>DeleteVaultAccessPolicy
- glacier:GetVaultLock
- glacier:DescribeVault
- glacier:SetVaultNotifications
- glacier:UploadMultipartPart
- glacier>AddTagsToVault
- glacier:InitiateMultipartUpload
- glacier:AbortMultipartUpload
- glacier>ListProvisionedCapacity

- `glacier:SetVaultAccessPolicy`
- `glacier>ListMultipartUploads`
- `glacier:InitiateVaultLock`
- `glacier>DeleteVaultNotifications`
- `glacier:CompleteVaultLock`
- `glacier:InitiateJob`
- `glacier>ListTagsForVault`
- `glacier>ListParts`
- `glacier>DeleteVault`
- `glacier>DeleteArchive`
- `glacier:GetVaultNotifications`
- `glacier:GetVaultAccessPolicy`
- `glacier:RemoveTagsFromVault`
- `glacier:CompleteMultipartUpload`
- `glacier:SetDataRetrievalPolicy`
- `glacier:UploadArchive`
- `glacier:AbortVaultLock`
- `glacier>ListVaults`
- `glacier>ListJobs`

Condition context keys for Amazon Glacier

Amazon Glacier has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `glacier:ArchiveAgeInDays`
- `glacier:ResourceTag/`

Actions and Condition Context Keys for AWS Glue

AWS Glue (service prefix: `glue`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Glue

For information about using the following AWS Glue API actions in an IAM policy, see [Overview of Managing Access Permissions to Your AWS Glue Resources](#) in the *AWS Glue Developer Guide*.

- `glue:BatchGetPartition`
- `glue:StartTrigger`
- `glue:GetConnection`
- `glue:GetTableVersions`
- `glue>CreateClassifier`
- `glue:GetCrawlers`
- `glue:UpdateClassifier`
- `glue:GetJobs`
- `glue:BatchDeleteTable`

- `glue:StartCrawler`
- `glue:BatchCreatePartition`
- `glue:DeleteClassifier`
- `glue:GetDataflowGraph`
- `glue>CreateJob`
- `glue:StopTrigger`
- `glue:StopCrawlerSchedule`
- `glue:UpdateConnection`
- `glue:GetPartition`
- `glue:CreateTrigger`
- `glue:UpdatePartition`
- `glue:DeleteTable`
- `glue:GetDevEndpoint`
- `glue:GetTable`
- `glue:GetDevEndpoints`
- `glue:GetMapping`
- `glue:DeletePartition`
- `glue:CreateUserDefinedFunction`
- `glue:DeleteUserDefinedFunction`
- `glue:GetTables`
- `glue:ImportCatalogToGlue`
- `glue:GetDatabase`
- `glue:ResetJobBookmark`
- `glue:GetCrawler`
- `glue>CreateConnection`
- `glue:DeleteCrawler`
- `glue:GetPlan`
- `glue:BatchDeletePartition`
- `glue:UpdateTrigger`
- `glue:CreateTable`
- `glue:UpdateDatabase`
- `glue:GetTriggers`
- `glue:GetJob`
- `glue:DeleteTrigger`
- `glue:GetTrigger`
- `glue:GetJobRun`
- `glue:CreateDevEndpoint`
- `glue:GetPartitions`
- `glue:UpdateJob`
- `glue:GetConnections`
- `glue:CreatePartition`
- `glue: GetUserDefinedFunctions`
- `glue:CreateDatabase`
- `glue:GetClassifiers`
- `glue:CreateScript`

- `glue:DeleteJob`
- `glue:StartCrawlerSchedule`
- `glue:GetJobRuns`
- `glue:BatchDeleteConnection`
- `glue:StopCrawler`
- `glue:DeleteDevEndpoint`
- `glue:DeleteConnection`
- `glue:GetCrawlerMetrics`
- `glue:GetCatalogImportStatus`
- `glue:GetClassifier`
- `glue:UpdateDevEndpoint`
- `glue:UpdateTable`
- `glue:UpdateUserDefinedFunction`
- `glue:GetDatabases`
- `glue:StartJobRun`
- `glue:DeleteDatabase`
- `glue:GetUserDefinedFunction`
- `glue>CreateCrawler`
- `glue:UpdateCrawler`

Condition context keys for AWS Glue

For information about using the following conditions in an IAM policy, see [Some Topic](#) in the .

AWS Glue has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Greengrass

AWS Greengrass (service prefix: greengrass) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Greengrass

- `greengrass:CreateResourceDefinitionVersion`
- `greengrass:DeleteCoreDefinition`
- `greengrass:GetGroupVersion`
- `greengrass:AssociateRoleToGroup`
- `greengrass:GetCoreDefinition`
- `greengrass>ListResourceDefinitionVersions`
- `greengrass:CreateSoftwareUpdateJob`
- `greengrass:DisassociateServiceRoleFromAccount`
- `greengrass:CreateGroupCertificateAuthority`
- `greengrass:UpdateResourceDefinition`
- `greengrass:GetLoggerDefinitionVersion`
- `greengrass:CreateResourceDefinition`
- `greengrass:DeleteResourceDefinition`

- greengrass>CreateFunctionDefinitionVersion
- greengrass>UpdateDeviceDefinition
- greengrass>ListCoreDefinitions
- greengrass>ListDeviceDefinitionVersions
- greengrass>GetAssociatedRole
- greengrass>ListGroupCertificateAuthorities
- greengrass>GetSubscriptionDefinition
- greengrass>CreateSubscriptionDefinitionVersion
- greengrass>CreateSubscriptionDefinition
- greengrass>GetLoggerDefinition
- greengrass>ListDeployments
- greengrass>CreateDeployment
- greengrass>GetGroup
- greengrass>CreateDeviceDefinition
- greengrass>GetGroupCertificateAuthority
- greengrass>CreateFunctionDefinition
- greengrass>CreateGroupVersion
- greengrass>GetResourceDefinition
- greengrass>CreateDeviceDefinitionVersion
- greengrass>DeleteDeviceDefinition
- greengrass>ListFunctionDefinitions
- greengrass>ListLoggerDefinitions
- greengrass>GetFunctionDefinitionVersion
- greengrass>UpdateSubscriptionDefinition
- greengrass>DeleteLoggerDefinition
- greengrass>ListLoggerDefinitionVersions
- greengrass>GetDeviceDefinition
- greengrass>DeleteFunctionDefinition
- greengrass>ListGroups
- greengrass>GetResourceDefinitionVersion
- greengrass>GetSubscriptionDefinitionVersion
- greengrass>GetFunctionDefinition
- greengrass>DeleteGroup
- greengrass>UpdateCoreDefinition
- greengrass>DisassociateRoleFromGroup
- greengrass>ListDeviceDefinitions
- greengrass>CreateCoreDefinition
- greengrass>GetConnectivityInfo
- greengrass>UpdateGroupCertificateConfiguration
- greengrass>ListResourceDefinitions
- greengrass>GetDeviceDefinitionVersion
- greengrass>AssociateServiceRoleToAccount
- greengrass>GetGroupCertificateConfiguration
- greengrass>ResetDeployments
- greengrass>GetServiceRoleForAccount

- `greengrass:GetCoreDefinitionVersion`
- `greengrass>ListFunctionDefinitionVersions`
- `greengrass>ListGroupVersions`
- `greengrass>CreateGroup`
- `greengrass:UpdateLoggerDefinition`
- `greengrass>CreateLoggerDefinitionVersion`
- `greengrass>CreateLoggerDefinition`
- `greengrass>ListSubscriptionDefinitions`
- `greengrass>ListSubscriptionDefinitionVersions`
- `greengrass>ListCoreDefinitionVersions`
- `greengrass>DeleteSubscriptionDefinition`
- `greengrass:UpdateConnectivityInfo`
- `greengrass:UpdateFunctionDefinition`
- `greengrass>CreateCoreDefinitionVersion`
- `greengrass:GetDeploymentStatus`
- `greengrass:UpdateGroup`

Condition context keys for AWS Greengrass

AWS Greengrass has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon GuardDuty

Amazon GuardDuty (service prefix: `guardduty`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon GuardDuty

- `guardduty:UpdateIPSet`
- `guardduty:GetMasterAccount`
- `guardduty:UpdateFindingsFeedback`
- `guardduty>CreateMembers`
- `guardduty:UpdateDetector`
- `guardduty>DeleteDetector`
- `guardduty:ArchiveFindings`
- `guardduty>ListInvitations`
- `guardduty>DeleteIPSet`
- `guardduty:GetInvitationsCount`
- `guardduty>ListFindings`
- `guardduty:GetIPSet`
- `guardduty:UpdateThreatIntelSet`
- `guardduty:GetDetector`
- `guardduty>CreateSampleFindings`
- `guardduty>DeleteThreatIntelSet`
- `guardduty>DeleteInvitations`
- `guardduty:GetThreatIntelSet`

- `guardduty:AcceptInvitation`
- `guardduty>CreateIPSet`
- `guardduty>ListMembers`
- `guardduty:DisassociateMembers`
- `guardduty:InviteMembers`
- `guardduty:GetMembers`
- `guardduty>ListDetectors`
- `guardduty>ListIPSets`
- `guardduty>CreateThreatIntelSet`
- `guardduty:GetFindings`
- `guardduty:UnarchiveFindings`
- `guardduty:StopMonitoringMembers`
- `guardduty:GetFindingsStatistics`
- `guardduty:DeclineInvitations`
- `guardduty>DeleteMembers`
- `guardduty>ListThreatIntelSets`
- `guardduty:StartMonitoringMembers`
- `guardduty>CreateDetector`
- `guardduty:DisassociateFromMasterAccount`

Condition context keys for Amazon GuardDuty

Amazon GuardDuty has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Health APIs and Notifications

AWS Health APIs and Notifications (service prefix: `health`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Health APIs and Notifications

For information about using the following AWS Health API actions in an IAM policy, see [Controlling Access to AWS Personal Health Dashboard and AWS Health](#) in the *AWS Health User Guide*.

- `health:DescribeEvents`
- `health:DescribeEventDetails`
- `health:DescribeEntityAggregates`
- `health:DescribeEventTypes`
- `health:DescribeAffectedEntities`
- `health:DescribeEventAggregates`

Condition context keys for AWS Health APIs and Notifications

AWS Health APIs and Notifications has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Identity And Access Management

Identity And Access Management (service prefix: iam) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Identity And Access Management

For information about controlling access to IAM managed policies, see [Controlling Access to Managed Policies](#) in the IAM User Guide guide. For information about controlling who can manage users and groups, see [Permissions for Administering IAM Users, Groups, and Credentials](#) in the *IAM User Guide* guide.

- [iam:ResetServiceSpecificCredential](#)
- [iam:UpdateServiceSpecificCredential](#)
- [iam:EnableMFADevice](#)
- [iam>ListUsers](#)
- [iam>CreateServiceLinkedRole](#)
- [iam>ListRolePolicies](#)
- [iam:PutRolePolicy](#)
- [iam>DeleteAccountPasswordPolicy](#)
- [iam>ListSSHPublicKeys](#)
- [iam>ListAttachedUserPolicies](#)
- [iam:AddUserToGroup](#)
- [iam:UpdateAssumeRolePolicy](#)
- [iam>ListEntitiesForPolicy](#)
- [iam>DeleteServiceLinkedRole](#)
- [iam>ListServerCertificates](#)
- [iam>ListAccountAliases](#)
- [iam:PassRole](#) - this is an IAM policy permission only, not an API action that can be called.
- [iam:GetContextKeysForCustomPolicy](#)
- [iam>ListInstanceProfiles](#)
- [iam>ListMFADevices](#)
- [iam:UploadSigningCertificate](#)
- [iam>DeleteRole](#)
- [iam:SimulateCustomPolicy](#)
- [iam>CreateServiceSpecificCredential](#)
- [iam:UpdateRoleDescription](#)
- [iam:ResyncMFADevice](#)
- [iam:AttachGroupPolicy](#)
- [iam>ListAttachedRolePolicies](#)
- [iam>ListPoliciesGrantingServiceAccess](#) - this is an IAM policy permission only, not an API action that can be called.
- [iamGetInstanceProfile](#)
- [iam:UpdateAccessKey](#)
- [iam:AddClientIDToOpenIDConnectProvider](#)
- [iam>ListGroupPolicies](#)
- [iam>DeleteOpenIDConnectProvider](#)

- [iam>CreateInstanceProfile](#)
- [iam:PutUserPolicy](#)
- [iam:ChangePassword](#)
- [iam:GenerateServiceLastAccessedDetails](#) - this is an IAM policy permission only, not an API action that can be called.
- [iam>CreateOpenIDConnectProvider](#)
- [iam:GetOpenIDConnectProvider](#)
- [iam>DeleteGroup](#)
- [iam>DeleteRolePolicy](#)
- [iam>ListServiceSpecificCredentials](#)
- [iam>ListRoles](#)
- [iam>CreateSAMLProvider](#)
- [iam>ListPolicyVersions](#)
- [iam>DeleteSSHPublicKey](#)
- [iam>CreateGroup](#)
- [iam>CreateUser](#)
- [iam>ListAccessKeys](#)
- [iam:UploadServerCertificate](#)
- [iam:GetRole](#)
- [iam:UploadSSHPublicKey](#)
- [iam:RemoveRoleFromInstanceProfile](#)
- [iam:UpdateSigningCertificate](#)
- [iam>DeleteLoginProfile](#)
- [iam:UpdateUser](#)
- [iam>ListVirtualMFADevices](#)
- [iam:GetSAMLProvider](#)
- [iam:AttachRolePolicy](#)
- [iam:UpdateAccountPasswordPolicy](#)
- [iam>CreatePolicy](#)
- [iam>DeleteServiceSpecificCredential](#)
- [iam:GetServiceLinkedRoleDeletionStatus](#)
- [iam:GetGroupPolicy](#)
- [iam:GetServiceLastAccessedDetailsWithEntities](#)
- [iam:DetachUserPolicy](#)
- [iam:GetLoginProfile](#)
- [iam>DeleteUserPolicy](#)
- [iam:UpdateLoginProfile](#)
- [iam:GetPolicyVersion](#)
- [iam:AddRoleToInstanceProfile](#)
- [iam:UpdateServerCertificate](#)
- [iam:DeactivateMFADevice](#)
- [iam:GetAccountPasswordPolicy](#)
- [iam:GetUser](#)
- [iam>DeleteVirtualMFADevice](#)
- [iam>DeletePolicyVersion](#)

- `iam:GetServiceLastAccessedDetails` - this is an IAM policy permission only, not an API action that can be called.
- `iam:RemoveUserFromGroup`
- `iam:AttachUserPolicy`
- `iam:UpdateOpenIDConnectProviderThumbprint`
- `iam:GetAccessKeyLastUsed`
- `iam:DeleteGroupPolicy`
- `iam:DeleteAccountAlias`
- `iam:ListGroup`
- `iam:UpdateSSHPublicKey`
- `iam>CreateAccessKey`
- `iam:DetachRolePolicy`
- `iam:GetSSHPublicKey`
- `iam>ListAttachedGroupPolicies`
- `iam:CreateAccountAlias`
- `iam:DeleteSigningCertificate`
- `iam>ListGroupsForUser`
- `iam>ListOpenIDConnectProviders`
- `iam:UpdateSAMLProvider`
- `iam>ListInstanceProfilesForRole`
- `iam:CreateVirtualMFADevice`
- `iam:ListGroup`
- `iam:DeleteUser`
- `iam:GetAccountAuthorizationDetails`
- `iam:DeletePolicy`
- `iam>ListSigningCertificates`
- `iam:PutGroupPolicy`
- `iam:RemoveClientIDFromOpenIDConnectProvider`
- `iam>ListPolicies`
- `iam:GetContextKeysForPrincipalPolicy`
- `iam:GenerateCredentialReport`
- `iam:SetDefaultPolicyVersion`
- `iam:CreateRole`
- `iam:CreatePolicyVersion`
- `iam:GetAccountSummary`
- `iam:GetServerCertificate`
- `iam:DetachGroupPolicy`
- `iam:DeleteSAMLProvider`
- `iam: GetUserPolicy`
- `iam:GetCredentialReport`
- `iam:DeleteAccessKey`
- `iam:DeleteServerCertificate`
- `iam>ListUserPolicies`
- `iam>ListSAMLProviders`
- `iam:GetRolePolicy`

- [iam>DeleteInstanceProfile](#)
- [iam>CreateLoginProfile](#)
- [iam:SimulatePrincipalPolicy](#)
- [iam:UpdateGroup](#)
- [iam:GetPolicy](#)

Condition context keys for Identity And Access Management

Identity And Access Management has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- [iam:PolicyARN](#)
- [iam:AWSServiceName](#)

Actions and Condition Context Keys for AWS Import Export Disk Service

AWS Import Export Disk Service (service prefix: importexport) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Import Export Disk Service

For information about using the following AWS Import/Export API actions in an IAM policy, see [Controlling Access to AWS Import/Export Jobs](#) in the *AWS Import/Export Developer Guide*.

- [importexport:GetShippingLabel](#)
- [importexport:UpdateJob](#)
- [importexport:CancelJob](#)
- [importexport>CreateJob](#)
- [importexport>ListJobs](#)
- [importexport:GetStatus](#)

Condition context keys for AWS Import Export Disk Service

AWS Import Export Disk Service has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Inspector

Amazon Inspector (service prefix: inspector) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Inspector

Amazon Inspector requires an IAM role to access your account to do its assessment. For more information, see [Setting Up Amazon Inspector](#) in the *Amazon Inspector User Guide*.

- [inspector:PreviewAgents](#)
- [inspector:DescribeResourceGroups](#)
- [inspector:RemoveAttributesFromFindings](#)

- `inspector:UnsubscribeFromEvent`
- `inspector>ListEventSubscriptions`
- `inspector:DescribeRulesPackages`
- `inspector>ListAssessmentTemplates`
- `inspector:StopAssessmentRun`
- `inspector:DescribeAssessmentRuns`
- `inspector:AddAttributesToFindings`
- `inspector:SubscribeToEvent`
- `inspector>ListAssessmentRunAgents`
- `inspector:DescribeCrossAccountAccessRole`
- `inspector:GetTelemetryMetadata`
- `inspector:UpdateAssessmentTarget`
- `inspector:StartAssessmentRun`
- `inspector>ListAssessmentRuns`
- `inspector>ListFindings`
- `inspector>DeleteAssessmentRun`
- `inspector:DescribeFindings`
- `inspector:DescribeAssessmentTargets`
- `inspector>ListAssessmentTargets`
- `inspector>CreateResourceGroup`
- `inspector:SetTagsForResource`
- `inspector:DescribeAssessmentTemplates`
- `inspector:RegisterCrossAccountAccessRole`
- `inspector>CreateAssessmentTemplate`
- `inspector>ListRulesPackages`
- `inspector>DeleteAssessmentTarget`
- `inspector>CreateAssessmentTarget`
- `inspector>DeleteAssessmentTemplate`
- `inspector>ListTagsForResource`

Condition context keys for Amazon Inspector

Amazon Inspector has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS IoT

AWS IoT (service prefix: `iot`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS IoT

For information about using IAM policies to grant permissions to run AWS IoT actions and access AWS IoT resources, see [Security and Identity for AWS IoT](#) in the *AWS IoT User Guide*.

- `iot:DescribeJob`
- `iot>ListPrincipalPolicies`
- `iot>ListJobExecutionsForThing`

- `iot>ListTopicRules`
- `iot>ListTargetsForPolicy`
- `iot>DeleteThing`
- `iot>UpdateStream`
- `iot>GetThingShadow` - this is an IAM policy permission only, not an API action that can be called.
- `iot>Subscribe` - this is an IAM policy permission only, not an API action that can be called.
- `iot>GetLoggingOptions`
- `iot>DeleteThingType`
- `iot>DeleteTopicRule`
- `iot>CancelJob`
- `iot>ListPrincipalThings`
- `iot>AttachPrincipalPolicy`
- `iot>CancelCertificateTransfer`
- `iot>CreateJob`
- `iot>UpdateThing`
- `iot>RegisterCertificate`
- `iot>UpdateIndexingConfiguration`
- `iot>CreateRoleAlias`
- `iot>Connect` - this is an IAM policy permission only, not an API action that can be called.
- `iot>DetachThingPrincipal`
- `iot>ListOutgoingCertificates`
- `iot>UpdateCACertificate`
- `iot>SetLoggingOptions`
- `iot>DeleteOTAUpdateJob`
- `iot>DeleteCertificate`
- `iot>DescribeIndex`
- `iot>ListThings`
- `iot>ListCertificates`
- `iot>CreateThingType`
- `iot>ListJobs`
- `iot>DescribeThingType`
- `iot>ListAttachedPolicies`
- `iot>CreateKeysAndCertificate`
- `iot>DescribeRoleAlias`
- `iot>ListThingTypes`
- `iot>EnableTopicRule`
- `iot>UpdateThingShadow` - this is an IAM policy permission only, not an API action that can be called.
- `iot>DeleteCACertificate`
- `iot>RegisterCACertificate`
- `iot>DeleteStream`
- `iot>ListThingPrincipals`
- `iot>ListPolicyPrincipals`
- `iot>UpdateRoleAlias`
- `iot>ListPolicyVersions`

- `iot:DeleteRoleAlias`
- `iot:DescribeJobExecution`
- `iot:SetDefaultAuthorizer`
- `iot:GetTopicRule`
- `iot:CreateTopicRule`
- `iot:CreateThing`
- `iot:Receive` - this is an IAM policy permission only, not an API action that can be called.
- `iot:ReplaceTopicRule`
- `iot:DeleteAuthorizer`
- `iot>ListStreams`
- `iot:CreatePolicy`
- `iot:TestAuthorization`
- `iot:TestInvokeAuthorizer`
- `iot:DescribeCertificate`
- `iot:AttachPolicy`
- `iot:GetPolicyVersion`
- `iot>ListOTAUpdateJobs`
- `iot:DeletePolicyVersion`
- `iot>ListJobExecutionsForJob`
- `iot:DescribeStream`
- `iot:AssociateTargetsWithJob`
- `iot:GetOTAUpdateJob`
- `iot:DescribeDefaultAuthorizer`
- `iot:DescribeAuthorizer`
- `iot:GetRegistrationCode`
- `iot:AcceptCertificateTransfer`
- `iot>ListAuthorizers`
- `iot:UpdateAuthorizer`
- `iot>CreateAuthorizer`
- `iot:ClearDefaultAuthorizer`
- `iot:DeletePolicy`
- `iot:AttachThingPrincipal`
- `iot:DetachPolicy`
- `iot>CreateOTAUpdateJob`
- `iot:GetEffectivePolicies`
- `iot>ListIndices`
- `iot:DeprecateThingType`
- `iot>ListPolicies`
- `iot:RejectCertificateTransfer`
- `iot:GetIndexingConfiguration`
- `iot>CreateCertificateFromCsr`
- `iot:DetachPrincipalPolicy`
- `iot:UpdateCertificate`
- `iot:DescribeCACertificate`
- `iot:SetDefaultPolicyVersion`

- `iot:Publish` - this is an IAM policy permission only, not an API action that can be called.
- `iot:CreatePolicyVersion`
- `iot:TransferCertificate`
- `iot:DescribeEndpoint`
- `iot:SearchIndex`
- `iot:CreateStream`
- `iot>ListCACertificates`
- `iot:DescribeThing`
- `iot:DeleteRegistrationCode`
- `iot>ListRoleAliases`
- `iot:DisableTopicRule`
- `iot:GetJobDocument`
- `iot>DeleteThingShadow` - this is an IAM policy permission only, not an API action that can be called.
- `iot>ListCertificatesByCA`
- `iot:GetPolicy`

Condition context keys for AWS IoT

AWS IoT has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS IoT Analytics

AWS IoT Analytics (service prefix: `iotanalytics`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS IoT Analytics

- `iotanalytics:DescribeChannel`
- `iotanalytics>CreateDatasetContent`
- `iotanalytics>ListChannels`
- `iotanalytics>CreatePipeline`
- `iotanalytics>ListPipelines`
- `iotanalytics>ListDatasets`
- `iotanalytics:DescribeDatastore`
- `iotanalytics>DeleteChannel`
- `iotanalytics>DeleteDatastore`
- `iotanalytics>CreateDataset`
- `iotanalytics>CreateDatastore`
- `iotanalytics:UpdateDataset`
- `iotanalytics>DeleteDatasetContent`
- `iotanalytics>DeletePipeline`
- `iotanalytics>CreateChannel`
- `iotanalytics>ListDatastores`
- `iotanalytics:GetDatasetContent`
- `iotanalytics:DescribeDataset`

- `iotanalytics:UpdatePipeline`
- `iotanalytics:DescribePipeline`
- `iotanalytics:DeleteDataset`

Condition context keys for AWS IoT Analytics

AWS IoT Analytics has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Key Management Service

AWS Key Management Service (service prefix: kms) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Key Management Service

For information about using the following AWS KMS API actions in an IAM resource policy attached to a AWS KMS key, see [Key Policies](#) in the *AWS Key Management Service Developer Guide*.

- `kms>ListGrants`
- `kms>GenerateDataKey`
- `kms>Decrypt`
- `kms>GenerateRandom`
- `kms>TagResource`
- `kms>CreateAlias`
- `kms>ListKeyPolicies`
- `kms>ListResourceTags`
- `kms>CreateGrant`
- `kms>RevokeGrant`
- `kms>GetKeyPolicy`
- `kms>ListKeys`
- `kms>ListRetirableGrants`
- `kms>PutKeyPolicy`
- `kms>ListAliases`
- `kms>CancelKeyDeletion`
- `kms>DisableKey`
- `kms>DeleteAlias`
- `kms>DescribeKey`
- `kms>ImportKeyMaterial`
- `kms>UpdateKeyDescription`
- `kms>Encrypt`
- `kms>GetKeyRotationStatus`
- `kms:ReEncryptFrom` - this is an IAM policy permission only, not an API action that can be called.
- `kms>DeleteImportedKeyMaterial`
- `kms:ReEncryptTo` - this is an IAM policy permission only, not an API action that can be called.
- `kms>DisableKeyRotation`

- `kms:UpdateAlias`
- `kms:UntagResource`
- `kms:RetireGrant`
- `kms:EnableKey`
- `kms:GenerateDataKeyWithoutPlaintext`
- `kms:EnableKeyRotation`
- `kms:ScheduleKeyDeletion`
- `kms:GetParametersForImport`
- `kms>CreateKey`

Condition context keys for AWS Key Management Service

For more information about using AWS KMS condition keys, see [Using Policy Conditions with AWS KMS](#).

AWS Key Management Service has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Kinesis

Amazon Kinesis (service prefix: `kinesis`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Kinesis

For information about using the following Kinesis API actions in an IAM policy, see [Actions for Kinesis](#) in the *Amazon Kinesis Developer Guide*.

- `kinesis:AddTagsToStream`
- `kinesis>ListStreams`
- `kinesis:EnableEnhancedMonitoring`
- `kinesis:DecreaseStreamRetentionPeriod`
- `kinesis:SplitShard`
- `kinesis>ListTagsForStream`
- `kinesis>CreateStream`
- `kinesis:DescribeStream`
- `kinesis:PutRecord`
- `kinesis:GetRecords`
- `kinesis:IncreaseStreamRetentionPeriod`
- `kinesis:GetShardIterator`
- `kinesis:DisableEnhancedMonitoring`
- `kinesis:DescribeLimits`
- `kinesis:PutRecords`
- `kinesis:RemoveTagsFromStream`
- `kinesis>DeleteStream`
- `kinesis:UpdateShardCount`
- `kinesis:MergeShards`

Condition context keys for Amazon Kinesis

Amazon Kinesis has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Kinesis Analytics

Amazon Kinesis Analytics (service prefix: `kinesisanalytics`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Kinesis Analytics

For information about using the following Amazon Kinesis Data Analytics API actions in an IAM policy, see [Granting Amazon Kinesis Data Analytics Permissions to Access Streaming Sources \(Creating an IAM Role\)](#) in the *Amazon Kinesis Data Analytics Developer Guide*.

- `kinesisanalytics:DescribeApplication`
- `kinesisanalytics:AddApplicationInput`
- `kinesisanalytics:AddApplicationOutput`
- `kinesisanalytics:DeleteApplicationOutput`
- `kinesisanalytics:StartApplication`
- `kinesisanalytics:DeleteApplicationReferenceDataSource`
- `kinesisanalytics:AddApplicationReferenceDataSource`
- `kinesisanalytics:DiscoverInputSchema`
- `kinesisanalytics>CreateApplication`
- `kinesisanalytics>ListApplications`
- `kinesisanalytics:StopApplication`
- `kinesisanalytics:DeleteApplication`
- `kinesisanalytics:UpdateApplication`

Condition context keys for Amazon Kinesis Analytics

Amazon Kinesis Analytics has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Kinesis Firehose

Amazon Kinesis Firehose (service prefix: `firehose`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Kinesis Firehose

For information about using IAM policies to grant permissions to run Kinesis Firehose actions and access Kinesis Firehose resources, see [Controlling Access with Kinesis Firehose](#) in the *Amazon Kinesis Firehose Developer Guide*.

- `firehose:PutRecordBatch`
- `firehose>CreateDeliveryStream`
- `firehose>ListDeliveryStreams`
- `firehose:DescribeDeliveryStream`
- `firehose:DeleteDeliveryStream`

- `firehose:PutRecord`
- `firehose:UpdateDestination`

Condition context keys for Amazon Kinesis Firehose

Amazon Kinesis Firehose has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Kinesis Video Streams

Amazon Kinesis Video Streams (service prefix: `kinesisvideo`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Kinesis Video Streams

- `kinesisvideo:PutMedia`
- `kinesisvideo>ListStreams`
- `kinesisvideo:GetMediaForFragmentList`
- `kinesisvideo>ListTagsForStream`
- `kinesisvideo:GetMedia`
- `kinesisvideo>CreateStream`
- `kinesisvideo:UpdateStream`
- `kinesisvideo:DescribeStream`
- `kinesisvideo:UntagStream`
- `kinesisvideo>ListFragments`
- `kinesisvideo>DeleteStream`
- `kinesisvideo:TagStream`
- `kinesisvideo:UpdateDataRetention`
- `kinesisvideo:GetDataEndpoint`

Condition context keys for Amazon Kinesis Video Streams

Amazon Kinesis Video Streams has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Lambda

AWS Lambda (service prefix: `lambda`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Lambda

For information about using the following Lambda API actions in an IAM policy, see [Permission Model](#) in the *AWS Lambda Developer Guide*.

- `lambda>ListEventSourceMappings`
- `lambda:TagResource`
- `lambda:AddPermission`

- `lambda>ListVersionsByFunction`
- `lambda>CreateFunction`
- `lambda>CreateAlias`
- `lambda>DeleteEventSourceMapping`
- `lambda.PutFunctionConcurrency`
- `lambda.RemovePermission`
- `lambda>CreateEventSourceMapping`
- `lambda.UpdateFunctionCode`
- `lambda.UpdateEventSourceMapping`
- `lambda.InvokeAsync`
- `lambda.GetEventSourceMapping`
- `lambda.EnableReplication`
- `lambda.PublishVersion`
- `lambda>ListAliases`
- `lambda_InvokeFunction` - this is an IAM policy permission only, not an API action that can be called.
- `lambda>DeleteAlias`
- `lambda.GetAccountSettings`
- `lambda>ListFunctions`
- `lambda>DeleteFunctionConcurrency`
- `lambda.UpdateAlias`
- `lambdaUntagResource`
- `lambda>ListTags`
- `lambdaGetFunctionConfiguration`
- `lambdaDeleteFunction`
- `lambdaGetAlias`
- `lambdaUpdateFunctionConfiguration`
- `lambdaGetFunction`
- `lambdaGetPolicy`

Condition context keys for AWS Lambda

AWS Lambda has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Lex

Amazon Lex (service prefix: `lex`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Lex

For information about using the following Amazon Lex API actions in an IAM policy, see [Authentication and Access Control for Amazon Lex](#) in the *Amazon Lex Developer Guide*.

- `lexPutSlotType`
- `lexGetBotChannelAssociation`
- `lexGetIntent`

- `lex>DeleteIntentVersion`
- `lex>CreateIntentVersion`
- `lex>GetBuiltInIntents`
- `lex>GetSlotTypeVersions`
- `lex>GetBots`
- `lex>GetIntents`
- `lex>GetSlotTypes`
- `lex>GetBot`
- `lex>GetBotAlias`
- `lex>DeleteBotAlias`
- `lex>PostContent`
- `lex>CreateBotVersion`
- `lex>PutIntent`
- `lex>GetIntentVersions`
- `lex>DeleteIntent`
- `lex>GetBuiltInSlotTypes`
- `lex>GetBotVersions`
- `lex>DeleteBotVersion`
- `lex>DeleteSlotType`
- `lex>GetUtterancesView`
- `lex>GetBotAliases`
- `lex>DeleteUtterances`
- `lex>DeleteBot`
- `lex>GetBuiltInIntent`
- `lex>GetSlotType`
- `lex>PutBot`
- `lex>CreateSlotTypeVersion`
- `lex>GetBotChannelAssociations`
- `lex>PostText`
- `lex>DeleteBotChannelAssociation`
- `lex>PutBotAlias`
- `lex>DeleteSlotTypeVersion`

Condition context keys for Amazon Lex

Amazon Lex has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `lex:associatedIntents`
- `lex:associatedSlotTypes`
- `lex:channelType`

Actions and Condition Context Keys for Amazon Lightsail

Amazon Lightsail (service prefix: `lightsail`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Lightsail

For more information about Amazon Lightsail, see the [Amazon LightSail documentation](#).

- [lightsail:GetKeyPair](#)
- [lightsail:GetOperation](#)
- [lightsail>DeleteKeyPair](#)
- [lightsail:ReleaseStaticIp](#)
- [lightsail:RebootInstance](#)
- [lightsail>DeleteInstanceSnapshot](#)
- [lightsail:AttachStaticIp](#)
- [lightsail:OpenInstancePublicPorts](#)
- [lightsail:ImportKeyPair](#)
- [lightsail:GetKeyPairs](#)
- [lightsail:PeerVpc](#)
- [lightsail>CreateInstancesFromSnapshot](#)
- [lightsail:GetStaticIp](#)
- [lightsail:GetInstancePortStates](#)
- [lightsail:GetInstances](#)
- [lightsail:GetOperations](#)
- [lightsail:GetInstanceSnapshots](#)
- [lightsail:UnpeerVpc](#)
- [lightsail>CreateInstanceSnapshot](#)
- [lightsail:IsVpcPeered](#)
- [lightsail>DeleteDomainEntry](#)
- [lightsail:StopInstance](#)
- [lightsail:GetDomain](#)
- [lightsail:GetBlueprints](#)
- [lightsail:GetInstanceMetricData](#)
- [lightsail:GetStaticIps](#)
- [lightsail:DetachStaticIp](#)
- [lightsail:CloseInstancePublicPorts](#)
- [lightsail:GetInstanceAccessDetails](#)
- [lightsail:GetInstance](#)
- [lightsail:GetRegions](#)
- [lightsail>DeleteDomain](#)
- [lightsail>CreateKeyPair](#)
- [lightsail:DownloadDefaultKeyPair](#)
- [lightsail>CreateDomain](#)
- [lightsail:GetInstanceState](#)
- [lightsail:GetActiveNames](#)
- [lightsail:GetOperationsForResource](#)
- [lightsail:GetInstanceSnapshot](#)
- [lightsail>CreateInstances](#)
- [lightsail:UpdateDomainEntry](#)
- [lightsail:AllocateStaticIp](#)

- `lightsail>CreateDomainEntry`
- `lightsail:GetDomains`
- `lightsail:StartInstance`
- `lightsail:GetBundles`
- `lightsail>DeleteInstance`

Condition context keys for Amazon Lightsail

Amazon Lightsail has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Machine Learning

Amazon Machine Learning (service prefix: `machinelearning`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Machine Learning

For information about using the following Amazon ML API actions in an IAM policy, see [Permission Model](#) in the *Amazon Machine Learning Developer Guide*.

- `machinelearning>CreateMLModel`
- `machinelearning>Predict`
- `machinelearning>DeleteRealtimeEndpoint`
- `machinelearning>GetDataSource`
- `machinelearning>CreateEvaluation`
- `machinelearning>DeleteMLModel`
- `machinelearning>UpdateBatchPrediction`
- `machinelearning>CreateBatchPrediction`
- `machinelearning>CreateDataSourceFromRedshift`
- `machinelearning>DescribeDataSources`
- `machinelearning>DeleteBatchPrediction`
- `machinelearning>GetMLModel`
- `machinelearning>GetEvaluation`
- `machinelearning>DescribeBatchPredictions`
- `machinelearning>UpdateMLModel`
- `machinelearning>DescribeTags`
- `machinelearning>GetBatchPrediction`
- `machinelearning>CreateDataSourceFromS3`
- `machinelearning>DeleteDataSource`
- `machinelearning>UpdateEvaluation`
- `machinelearning>CreateRealtimeEndpoint`
- `machinelearning>DeleteEvaluation`
- `machinelearning>CreateDataSourceFromRDS`
- `machinelearning>DescribeMLModels`
- `machinelearning>UpdateDataSource`
- `machinelearning>AddTags`

- `machinelearning:DeleteTags`
- `machinelearning:DescribeEvaluations`

Condition context keys for Amazon Machine Learning

Amazon Machine Learning has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Manage Amazon API Gateway

Manage Amazon API Gateway (service prefix: apigateway) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Manage Amazon API Gateway

For more information about controlling access to API Gateway, see [User Access Permissions for Amazon API Gateway](#) in the *API Gateway Developer Guide*.

- `apigateway:HEAD`
- `apigateway:DELETE`
- `apigateway:POST`
- `apigateway:GET`
- `apigateway:OPTIONS`
- `apigateway:PUT`
- `apigateway:PATCH`

Condition context keys for Manage Amazon API Gateway

Manage Amazon API Gateway has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Marketplace

AWS Marketplace (service prefix: aws-marketplace) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Marketplace

For information about using the following MarketPlace Subscription API actions in an IAM policy, see [Permission Details for AWS Marketplace Subscriptions](#) in the *AWS Marketplace User Guide*.

- `aws-marketplace:BatchMeterUsage`
- `aws-marketplace:MeterUsage`
- `aws-marketplace:ResolveCustomer`
- `aws-marketplace:Subscribe`
- `aws-marketplace:Unsubscribe`
- `aws-marketplace:ViewSubscriptions`

Condition context keys for AWS Marketplace

AWS Marketplace has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Marketplace Management Portal

AWS Marketplace Management Portal (service prefix: aws-marketplace-management) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Marketplace Management Portal

For information about using the following MarketPlace Management API actions in an IAM policy, see [Permission Details for AWS Marketplace Management Portal](#) in the *AWS Marketplace User Guide*.

- `aws-marketplace-management:uploadFiles`
- `aws-marketplace-management:viewReports`
- `aws-marketplace-management:viewMarketing`
- `aws-marketplace-management:viewSupport`

Condition context keys for AWS Marketplace Management Portal

AWS Marketplace Management Portal has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Mechanical Turk

Amazon Mechanical Turk (service prefix: mechanicturk) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Mechanical Turk

- `mechanicalturk:ApproveAssignment`
- `mechanicalturk:GetQualificationRequests`
- `mechanicalturk:RevokeQualification`
- `mechanicalturk:NotifyWorkers`
- `mechanicalturk:GetHITsForQualificationType`
- `mechanicalturk:SetHITTypeNotification`
- `mechanicalturk:UpdateQualificationType`
- `mechanicalturk>CreateHIT`
- `mechanicalturk:GetFileUploadURL`
- `mechanicalturk:RejectAssignment`
- `mechanicalturk:GetQualificationsForQualificationType`
- `mechanicalturk:ChangeHITTypeOfHIT`
- `mechanicalturk:GetHIT`
- `mechanicalturk:GetRequesterWorkerStatistic`
- `mechanicalturk:GetReviewableHITs`
- `mechanicalturk:GetRequesterStatistic`
- `mechanicalturk:SearchHITS`

- `mechanicalturk:GrantBonus`
- `mechanicalturk:ExtendHIT`
- `mechanicalturk>CreateQualificationType`
- `mechanicalturk:AssignQualification`
- `mechanicalturk:UpdateQualificationScore`
- `mechanicalturk:ApproveRejectedAssignment`
- `mechanicalturk:GetQualificationType`
- `mechanicalturk:SendTestEventNotification`
- `mechanicalturk:SearchQualificationTypes`
- `mechanicalturk:GetAssignment`
- `mechanicalturk:GetAccountBalance`
- `mechanicalturk:DisposeQualificationType`
- `mechanicalturk:RegisterHITType`
- `mechanicalturk:RejectQualificationRequest`
- `mechanicalturk:GetReviewResultsForHIT`
- `mechanicalturk:UnblockWorker`
- `mechanicalturk:GetBlockedWorkers`
- `mechanicalturk:DisposeHIT`
- `mechanicalturk:GetQualificationScore`
- `mechanicalturk:GetAssignmentsForHIT`
- `mechanicalturk:GrantQualification`
- `mechanicalturk:DisableHIT`
- `mechanicalturk:GetBonusPayments`
- `mechanicalturk:SetHITAsReviewing`
- `mechanicalturk:BlockWorker`
- `mechanicalturk:ForceExpireHIT`

Condition context keys for Amazon Mechanical Turk

Amazon Mechanical Turk has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Mechanical Turk Crowd

Amazon Mechanical Turk Crowd (service prefix: `crowd`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Mechanical Turk Crowd

- `crowd:GetTask`
- `crowd:PutTask`

Condition context keys for Amazon Mechanical Turk Crowd

Amazon Mechanical Turk Crowd has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Message Delivery Service

Amazon Message Delivery Service (service prefix: ec2messages) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Message Delivery Service

- `ec2messages:GetEndpoint`
- `ec2messages:AcknowledgeMessage`
- `ec2messages:SendReply`
- `ec2messages:FailMessage`
- `ec2messages:DeleteMessage`
- `ec2messages:GetMessages`

Condition context keys for Amazon Message Delivery Service

Amazon Message Delivery Service has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Migration Hub

AWS Migration Hub (service prefix: mgh) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Migration Hub

- `mgh:DisassociateCreatedArtifact`
- `mgh:DescribeApplicationState`
- `mgh:DescribeMigrationTask`
- `mgh:DeleteProgressUpdateStream`
- `mgh>ListMigrationTasks`
- `mgh>ListCreatedArtifacts`
- `mgh:AssociateCreatedArtifact`
- `mgh>ListProgressUpdateStreams`
- `mgh:DisassociateDiscoveredResource`
- `mgh:AssociateDiscoveredResource`
- `mgh:NotifyMigrationTaskState`
- `mgh:NotifyApplicationState`
- `mgh:ImportMigrationTask`
- `mgh:PutResourceAttributes`
- `mgh>ListDiscoveredResources`
- `mgh>CreateProgressUpdateStream`

Condition context keys for AWS Migration Hub

AWS Migration Hub has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Mobile Analytics

Amazon Mobile Analytics (service prefix: mobileanalytics) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Mobile Analytics

- `mobileanalytics:PutEvents`
- `mobileanalytics:GetReports`
- `mobileanalytics:GetFinancialReports`

Condition context keys for Amazon Mobile Analytics

Amazon Mobile Analytics has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Mobile Hub

AWS Mobile Hub (service prefix: mobilehub) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Mobile Hub

- `mobilehub:DescribeBundle`
- `mobilehub:ExportProject`
- `mobilehub>ListAvailableFeatures`
- `mobilehub:ImportProject`
- `mobilehub>ListProjects`
- `mobilehub>ListAvailableConnectors`
- `mobilehub:ExportBundle`
- `mobilehub>CreateProject`
- `mobilehub:GetProject`
- `mobilehub>DeleteProject`
- `mobilehub:UpdateProject`
- `mobilehub:GenerateProjectParameters`
- `mobilehub>ListAvailableRegions`
- `mobilehub:DeployToStage`
- `mobilehub:GetProjectSnapshot`
- `mobilehub>ListBundles`
- `mobilehub:SynchronizeProject`
- `mobilehub>CreateServiceRole`
- `mobilehub:VerifyServiceRole`

Condition context keys for AWS Mobile Hub

AWS Mobile Hub has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon MQ

Amazon MQ (service prefix: mq) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon MQ

- `mq>ListBrokers`
- `mq>CreateBroker`
- `mqUpdateBroker`
- `mqRebootBroker`
- `mqListUsers`
- `mqDescribeBroker`
- `mqCreateUser`
- `mqUpdateConfiguration`
- `mqDeleteUser`
- `mqCreateConfiguration`
- `mqDescribeUser`
- `mqDescribeConfiguration`
- `mqListConfigurations`
- `mqDescribeConfigurationRevision`
- `mqUpdateUser`
- `mqDeleteBroker`
- `mqListConfigurationRevisions`

Condition context keys for Amazon MQ

Amazon MQ has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS OpsWorks

AWS OpsWorks (service prefix: opsworks) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS OpsWorks

For information about using the following AWS OpsWorks API actions in an IAM policy, see the Action section in [Managing AWS OpsWorks Permissions by Attaching an IAM Policy](#) in the *AWS OpsWorks User Guide*.

- `opsworksUnassignVolume`
- `opsworksDescribeCommands`
- `opsworksDescribeApps`
- `opsworksSetLoadBasedAutoScaling`
- `opsworksStartStack`
- `opsworksCloneStack`
- `opsworksUnassignInstance`
- `opsworksDeregisterVolume`

- `opsworks:DescribeServiceErrors`
- `opsworks:UpdateElasticIp`
- `opsworks:CreateInstance`
- `opsworks:UpdateStack`
- `opsworks:AssociateElasticIp`
- `opsworks:DescribeStackProvisioningParameters`
- `opsworks:DescribeStackSummary`
- `opsworks:SetPermission`
- `opsworks>CreateStack`
- `opsworks:UpdateRdsDbInstance`
- `opsworks:DeleteStack`
- `opsworks:DescribeUserProfiles`
- `opsworks:DetachElasticLoadBalancer`
- `opsworks:RegisterElasticIp`
- `opsworks:StopInstance`
- `opsworks:RegisterRdsDbInstance`
- `opsworks:UpdateInstance`
- `opsworks:DescribeMyUserProfile`
- `opsworks:DescribeRaidArrays`
- `opsworks>CreateDeployment`
- `opsworks:GrantAccess`
- `opsworks:RegisterEcsCluster`
- `opsworks:DeleteLayer`
- `opsworks:UntagResource`
- `opsworks:DescribeInstances`
- `opsworks:UpdateApp`
- `opsworks:DescribeRdsDbInstances`
- `opsworks:StartInstance`
- `opsworks:DeleteUserProfile`
- `opsworks:DescribeTimeBasedAutoScaling`
- `opsworks:TagResource`
- `opsworks:DeleteApp`
- `opsworks:RebootInstance`
- `opsworks:GetHostnameSuggestion`
- `opsworks:RegisterVolume`
- `opsworks:UpdateLayer`
- `opsworks:UpdateUserProfile`
- `opsworks:DescribeElasticIps`
- `opsworks:CreateUserProfile`
- `opsworks:UpdateTimeBasedAutoScaling`
- `opsworks:DeregisterEcsCluster`
- `opsworks:UpdateMyUserProfile`
- `opsworks:AttachElasticLoadBalancer`
- `opsworks:DescribePermissions`
- `opsworks:DescribeLayers`

- [opsworks:DescribeStacks](#)
- [opsworks:DescribeVolumes](#)
- [opsworks:DescribeAgentVersions](#)
- [opsworks:RegisterInstance](#)
- [opsworks:AssignInstance](#)
- [opsworks:DeregisterInstance](#)
- [opsworks:DeregisterElasticIp](#)
- [opsworks:DeregisterRdsDbInstance](#)
- [opsworks:StopStack](#)
- [opsworks:AssignVolume](#)
- [opsworks:UpdateVolume](#)
- [opsworks>ListTags](#)
- [opsworks>CreateApp](#)
- [opsworks:DescribeLoadBasedAutoScaling](#)
- [opsworks:DisassociateElasticIp](#)
- [opsworks:DescribeDeployments](#)
- [opsworks:DescribeEcsClusters](#)
- [opsworks:DescribeElasticLoadBalancers](#)
- [opsworks>DeleteInstance](#)
- [opsworks>CreateLayer](#)

Condition context keys for AWS OpsWorks

AWS OpsWorks has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS OpsWorks Configuration Management

AWS OpsWorks Configuration Management (service prefix: opsworks-cm) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS OpsWorks Configuration Management

For information about using the following AWS OpsWorks for Chef Automate API actions in an IAM policy, see [Best Practices: Managing Permissions](#) in the *AWS OpsWorks User Guide*.

- [opsworks-cm:StartMaintenance](#)
- [opsworks-cm:UpdateServer](#)
- [opsworks-cm:DeleteServer](#)
- [opsworks-cm:DeleteBackup](#)
- [opsworks-cm:DescribeAccountAttributes](#)
- [opsworks-cm:DisassociateNode](#)
- [opsworks-cm:DescribeEvents](#)
- [opsworks-cm:DescribeServers](#)
- [opsworks-cm:RestoreServer](#)
- [opsworks-cm:DescribeBackups](#)

- `opsworks-cm:AssociateNode`
- `opsworks-cm:DescribeNodeAssociationStatus`
- `opsworks-cm:UpdateServerEngineAttributes`
- `opsworks-cm>CreateServer`
- `opsworks-cm>CreateBackup`

Condition context keys for AWS OpsWorks Configuration Management

AWS OpsWorks Configuration Management has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Organizations

AWS Organizations (service prefix: organizations) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Organizations

For additional information about using AWS Organizations actions in an IAM policy, see [Using Identity-Based Policies \(IAM Policies\) for AWS Organizations](#) in the *AWS Organizations User Guide*.

- `organizations>DeleteOrganizationalUnit`
- `organizations>CreatePolicy`
- `organizations>ListOrganizationalUnitsForParent`
- `organizations>DescribeCreateAccountStatus`
- `organizations>ListTargetsForPolicy`
- `organizations>ListAccounts`
- `organizations>DescribePolicy`
- `organizations>DeletePolicy`
- `organizations>DetachPolicy`
- `organizations>EnablePolicyType`
- `organizations>EnableAllFeatures`
- `organizations>DescribeOrganizationalUnit`
- `organizations>ListParents`
- `organizations>ListChildren`
- `organizations>UpdateOrganizationalUnit`
- `organizations>ListPolicies`
- `organizations>ListHandshakesForAccount`
- `organizations>AcceptHandshake`
- `organizations>AttachPolicy`
- `organizations>CreateOrganizationalUnit`
- `organizations>DisablePolicyType`
- `organizations>DeclineHandshake`
- `organizations>DescribeAccount`
- `organizations>ListPoliciesForTarget`
- `organizations>UpdatePolicy`
- `organizations>InviteAccountToOrganization`

- organizations:CreateAccount
- organizations:CancelHandshake
- organizations:DescribeHandshake
- organizations:CreateOrganization
- organizations>ListAccountsForParent
- organizations>ListHandshakesForOrganization
- organizations:LeaveOrganization
- organizations>ListRoots
- organizations:MoveAccount
- organizations>ListCreateAccountStatus
- organizations:RemoveAccountFromOrganization
- organizations:DescribeOrganization
- organizations>DeleteOrganization

Condition context keys for AWS Organizations

AWS Organizations has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Pinpoint

Amazon Pinpoint (service prefix: mobiletargeting) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Pinpoint

For information about using the following Amazon Pinpoint API actions in an IAM policy, see [Permissions in the Amazon Pinpoint Developer Guide](#).

- mobiletargeting:UpdateApplicationSettings
- mobiletargeting:GetSegments
- mobiletargeting:GetImportJob
- mobiletargeting:GetCampaign
- mobiletargeting>CreateImportJob
- mobiletargeting:GetGcmChannel
- mobiletargeting:UpdateGcmChannel
- mobiletargeting>DeleteApnsChannel
- mobiletargeting:UpdateSegment
- mobiletargeting:UpdateCampaign
- mobiletargeting:GetCampaignVersions
- mobiletargeting>DeleteGcmChannel
- mobiletargeting:GetSegmentVersion
- mobiletargeting:GetCampaignVersion
- mobiletargeting>CreateSegment
- mobiletargeting:GetSegment
- mobiletargeting:GetApnsChannel
- mobiletargeting:GetEndpoint
- mobiletargeting>CreateCampaign

- `mobiletargeting:GetCampaigns`
- `mobiletargeting:GetCampaignActivities`
- `mobiletargeting>DeleteSegment`
- `mobiletargeting>DeleteCampaign`
- `mobiletargeting:GetReports`
- `mobiletargeting:UpdateEndpoint`
- `mobiletargeting:GetSegmentImportJobs`
- `mobiletargeting:UpdateApnsChannel`
- `mobiletargeting:GetImportJobs`
- `mobiletargeting:GetApplicationSettings`
- `mobiletargeting:GetSegmentVersions`
- `mobiletargeting:UpdateEndpointsBatch`

Condition context keys for Amazon Pinpoint

Amazon Pinpoint has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Polly

Amazon Polly (service prefix: `polly`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Polly

For information about using the following Amazon Polly API actions in an IAM policy, see [Using Identity-Based Policies \(IAM Policies\) for Amazon Polly](#) in the *Amazon Polly Developer Guide*.

- `polly:GetLexicon`
- `polly:PutLexicon`
- `polly:DescribeVoices`
- `polly>ListLexicons`
- `polly:DeleteLexicon`
- `polly:SynthesizeSpeech`

Condition context keys for Amazon Polly

Amazon Polly has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Price List

AWS Price List (service prefix: `pricing`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Price List

- `pricing:GetAttributeValues`
- `pricing:GetProducts`

- `pricing:DescribeServices`

Condition context keys for AWS Price List

AWS Price List has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon RDS

Amazon RDS (service prefix: rds) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon RDS

For information about using the following Amazon RDS API actions in an IAM policy, see [Using AWS Identity and Access Management \(IAM\) to Manage Access to Amazon RDS Resources](#) in the *Amazon Relational Database Service User Guide*.

For the REST API for `DownloadCompleteDBLogFile`, see [DownloadCompleteDBLogFile](#).

- `rds:AuthorizeDBSecurityGroupIngress`
- `rds:DescribeDBInstances`
- `rds:CopyDBClusterSnapshot`
- `rds:DescribeCertificates`
- `rds:DescribePendingMaintenanceActions`
- `rds:RevokeDBSecurityGroupIngress`
- `rds:DownloadCompleteDBLogFile` - this is an IAM policy permission only, not an API action that can be called.
- `rds:DownloadDBLogFilePortion`
- `rds>CreateDBInstance`
- `rds:PromoteReadReplica`
- `rds:ResetDBParameterGroup`
- `rds:ModifyDBSnapshotAttribute`
- `rds:DescribeDBSnapshotAttributes`
- `rds:CreateDBClusterParameterGroup`
- `rds:FailoverDBCluster`
- `rds:DeleteDBCluster`
- `rds:ApplyPendingMaintenanceAction`
- `rds:CopyDBSnapshot`
- `rds:CopyDBParameterGroup`
- `rds:CreateDBClusterSnapshot`
- `rds:DescribeAccountAttributes`
- `rds:StopDBInstance`
- `rds:DeleteDBSnapshot`
- `rds:DescribeDBClusterParameterGroups`
- `rds:DeleteDBSecurityGroup`
- `rds:ModifyDBParameterGroup`
- `rds:CopyOptionGroup`
- `rds:ModifyDBSubnetGroup`

- `rds:DescribeEngineDefaultParameters`
- `rds:ModifyOptionGroup`
- `rds:DescribeDBParameters`
- `rds:DeleteEventSubscription`
- `rds>CreateDBCluster`
- `rds:RestoreDBClusterFromSnapshot`
- `rds:AddTagsToResource`
- `rds>ListTagsForResource`
- `rds:CreateEventSubscription`
- `rds:CreateDBSecurityGroup`
- `rds:CreateDBSnapshot`
- `rds:DeleteDBSubnetGroup`
- `rds:DescribeDBClusterSnapshotAttributes`
- `rds:CreateDBSubnetGroup`
- `rds:DescribeReservedDBInstances`
- `rds:DescribeEventCategories`
- `rds:DescribeDBClusters`
- `rds:DescribeOrderableDBInstanceOptions`
- `rds:RestoreDBInstanceToPointInTime`
- `rds:DescribeDBEngineVersions`
- `rds:DescribeOptionGroups`
- `rds:DeleteOptionGroup`
- `rds:ModifyDBInstance`
- `rds:DescribeDBSecurityGroups`
- `rds:RestoreDBInstanceFromDBSnapshot`
- `rds:DescribeDBLogFiles`
- `rds:ModifyEventSubscription`
- `rds>DeleteDBInstance`
- `rds:DeleteDBClusterParameterGroup`
- `rds:CreateDBInstanceReadReplica`
- `rds:RemoveSourceIdentifierFromSubscription`
- `rds:DescribeEventSubscriptions`
- `rds:PurchaseReservedDBInstancesOffering`
- `rds:CreateDBParameterGroup`
- `rds:ResetDBClusterParameterGroup`
- `rds:StartDBInstance`
- `rds:AddRoleToDBCluster`
- `rds:RemoveTagsFromResource`
- `rds:ModifyDBClusterSnapshotAttribute`
- `rds:AddSourceIdentifierToSubscription`
- `rds:DescribeEvents`
- `rds:DescribeDBSnapshots`
- `rds:ModifyDBClusterParameterGroup`
- `rds:DeleteDBParameterGroup`
- `rds:DescribeEngineDefaultClusterParameters`

- `rds:DeleteDBClusterSnapshot`
- `rds:RebootDBInstance`
- `rds:RestoreDBClusterToPointInTime`
- `rds:ModifyDBCluster`
- `rds:DescribeDBParameterGroups`
- `rds:DescribeDBSubnetGroups`
- `rds:DescribeDBClusterSnapshots`
- `rds>CreateOptionGroup`
- `rds:DescribeOptionGroupOptions`
- `rds:DescribeDBClusterParameters`
- `rds:DescribeReservedDBInstancesOfferings`

Condition context keys for Amazon RDS

For information about using the following Amazon RDS conditions in an IAM policy, see [Specifying Conditions in an IAM Policy for Amazon RDS](#) in the *Amazon Relational Database Service User Guide*.

Amazon RDS has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `Piops`
- `rds:DatabaseClass`
- `rds:DatabaseEngine`
- `rds:DatabaseName`
- `rds:MultiAz`
- `rds:Piops`
- `rds:StorageSize`
- `rds:Vpc`
- `rds:cluster-pg-tag`
- `rds:cluster-snapshot-tag`
- `rds:cluster-tag`
- `rds:db-tag`
- `rds:es-tag`
- `rds:og-tag`
- `rds:pg-tag`
- `rds:ri-tag`
- `rds:secgrp-tag`
- `rds:snapshot-tag`
- `rds:subgrp-tag`

Actions and Condition Context Keys for Amazon Redshift

Amazon Redshift (service prefix: `redshift`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Redshift

- `redshift:EnableLogging`

- `redshift:DescribeClusterVersions`
- `redshift:GetClusterCredentials`
- `redshift:DescribeHsmConfigurations`
- `redshift:ModifyCluster`
- `redshift:DescribeLoggingStatus`
- `redshift:EnableSnapshotCopy`
- `redshift:DeleteClusterParameterGroup`
- `redshift:DeleteClusterSnapshot`
- `redshift>CreateClusterSnapshot`
- `redshift:CancelQuerySession`
- `redshift:CreateHsmClientCertificate`
- `redshift:AuthorizeClusterSecurityGroupIngress`
- `redshift:DescribeClusters`
- `redshift:DescribeTags`
- `redshift:ModifyClusterSubnetGroup`
- `redshift:DeleteHsmClientCertificate`
- `redshift:PurchaseReservedNodeOffering`
- `redshift:DescribeResize`
- `redshift:DescribeClusterSubnetGroups`
- `redshift:RebootCluster`
- `redshift:CreateSnapshotCopyGrant`
- `redshift:DescribeClusterSecurityGroups`
- `redshift:CreateClusterSecurityGroup`
- `redshift:CreateClusterSubnetGroup`
- `redshift:RestoreFromClusterSnapshot`
- `redshift:RestoreTableFromClusterSnapshot`
- `redshift:DeleteEventSubscription`
- `redshift:CreateTags`
- `redshift:CopyClusterSnapshot`
- `redshift:ModifyClusterParameterGroup`
- `redshift:CreateCluster`
- `redshift:CreateEventSubscription`
- `redshift:ModifySnapshotCopyRetentionPeriod`
- `redshift:ViewQueriesInConsole`
- `redshift:DescribeEventCategories`
- `redshift:DeleteCluster`
- `redshift:AuthorizeSnapshotAccess`
- `redshift:DescribeClusterParameters`
- `redshift:DeleteClusterSubnetGroup`
- `redshift:DescribeClusterParameterGroups`
- `redshift:DescribeHsmClientCertificates`
- `redshift:RevokeSnapshotAccess`
- `redshift:DescribeSnapshotCopyGrants`
- `redshift:RotateEncryptionKey`
- `redshift:ModifyClusterIamRoles`

- [redshift:DescribeReservedNodes](#)
- [redshift:DeleteClusterSecurityGroup](#)
- [redshift:DescribeClusterSnapshots](#)
- [redshift:ModifyEventSubscription](#)
- [redshift:DescribeEventSubscriptions](#)
- [redshift:RevokeClusterSecurityGroupIngress](#)
- [redshift>CreateClusterParameterGroup](#)
- [redshift>DeleteHsmConfiguration](#)
- [redshift:DisableSnapshotCopy](#)
- [redshift:DescribeOrderableClusterOptions](#)
- [redshift:DeleteSnapshotCopyGrant](#)
- [redshift:DescribeReservedNodeOfferings](#)
- [redshift>CreateHsmConfiguration](#)
- [redshift:DescribeTableRestoreStatus](#)
- [redshift:DescribeEvents](#)
- [redshift:CreateClusterUser](#)
- [redshift:DescribeDefaultClusterParameters](#)
- [redshift:ResetClusterParameterGroup](#)
- [redshift:DisableLogging](#)
- [redshift:DeleteTags](#)
- [redshift:JoinGroup](#)

Condition context keys for Amazon Redshift

Amazon Redshift has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- [redshift:DbName](#)
- [redshift:DbUser](#)
- [redshift:DurationSeconds](#)

Actions and Condition Context Keys for Amazon Rekognition

Amazon Rekognition (service prefix: `rekognition`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Rekognition

For information about using the following Amazon Rekognition API actions in an IAM policy, see [Using Identity-Based Policies \(IAM Policies\) for Amazon Rekognition](#) in the *Amazon Rekognition Developer Guide*.

- [rekognition:GetPersonTracking](#)
- [rekognition:DeleteFaces](#)
- [rekognition:DetectText](#)
- [rekognition>ListFaces](#)
- [rekognition:StartPersonTracking](#)
- [rekognition:GetCelebrityRecognition](#)

- `rekognition:SearchFaces`
- `rekognition>CreateCollection`
- `rekognition:IndexFaces`
- `rekognition:GetContentModeration`
- `rekognition:StartFaceSearch`
- `rekognition>ListStreamProcessors`
- `rekognition>DeleteCollection`
- `rekognition:StartCelebrityRecognition`
- `rekognition:StartFaceDetection`
- `rekognition:StartContentModeration`
- `rekognition:GetFaceSearch`
- `rekognition:GetLabelDetection`
- `rekognition:GetCelebrityInfo`
- `rekognition:CompareFaces`
- `rekognition:DetectFaces`
- `rekognition>ListCollections`
- `rekognition:DetectLabels`
- `rekognition:DetectModerationLabels`
- `rekognition:StopStreamProcessor`
- `rekognition:StartStreamProcessor`
- `rekognition:DescribeStreamProcessor`
- `rekognition:SearchFacesByImage`
- `rekognition:StartLabelDetection`
- `rekognition>DeleteStreamProcessor`
- `rekognition>CreateStreamProcessor`
- `rekognition:RecognizeCelebrities`
- `rekognition:GetFaceDetection`

Condition context keys for Amazon Rekognition

Amazon Rekognition has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Resource Group Tagging API

Amazon Resource Group Tagging API (service prefix: tag) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Resource Group Tagging API

For information about using the following Resource Group Tagging API actions in an IAM policy, see [Obtaining](#).

- `tag:UntagResources`
- `tag:TagResources`
- `tag:GetResources`

- [tag:GetTagKeys](#)
- [tag:GetTagValues](#)

Condition context keys for Amazon Resource Group Tagging API

Amazon Resource Group Tagging API has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Resource Groups

AWS Resource Groups (service prefix: resource-groups) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Resource Groups

- [resource-groups:ListGroupResources](#)
- [resource-groups:ListGroupQuery](#)
- [resource-groups:SearchResources](#)
- [resource-groups:CreateGroup](#)
- [resource-groups>ListGroups](#)
- [resource-groups:Tag](#)
- [resource-groups>DeleteGroup](#)
- [resource-groups:UpdateGroup](#)
- [resource-groups:UpdateGroupQuery](#)

Condition context keys for AWS Resource Groups

AWS Resource Groups has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Route 53

Amazon Route 53 (service prefix: route53) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Route 53

For information about using the following Route 53 API actions in an IAM policy, see [Route 53 Actions](#) in the *Amazon Route 53 Developer Guide*.

- [route53:TestDNSAnswer](#)
- [route53>ListTagsForResources](#)
- [route53:GetHostedZone](#)
- [route53>ListResourceRecordSets](#)
- [route53>CreateHealthCheck](#)
- [route53:GetHealthCheckCount](#)
- [route53>ListGeoLocations](#)

- [route53:UpdateTrafficPolicyComment](#)
- [route53>ListTrafficPolicyInstances](#)
- [route53>ListTrafficPolicyInstancesByHostedZone](#)
- [route53:ChangeResourceRecordSets](#)
- [route53>CreateReusableDelegationSet](#)
- [route53:EnableDomainAutoRenew](#)
- [route53>CreateTrafficPolicy](#)
- [route53:UpdateTrafficPolicyInstance](#)
- [route53>DeleteHealthCheck](#)
- [route53>ListTrafficPolicyInstancesByPolicy](#)
- [route53>CreateHostedZone](#)
- [route53:GetTrafficPolicyInstance](#)
- [route53:GetGeoLocation](#)
- [route53>DeleteHostedZone](#)
- [route53>ListTrafficPolicyVersions](#)
- [route53>DeleteTrafficPolicyInstance](#)
- [route53>DeleteReusableDelegationSet](#)
- [route53:DisassociateVPCFromHostedZone](#)
- [route53>ListTrafficPolicies](#)
- [route53:DisableDomainAutoRenew](#)
- [route53:GetChange](#)
- [route53>ListHostedZones](#)
- [route53:AssociateVPCWithHostedZone](#)
- [route53:GetHealthCheck](#)
- [route53:GetHealthCheckStatus](#)
- [route53:GetHealthCheckLastFailureReason](#)
- [route53>CreateTrafficPolicyVersion](#)
- [route53>ListReusableDelegationSets](#)
- [route53:UpdateHealthCheck](#)
- [route53:GetCheckerIpRanges](#)
- [route53:GetHostedZoneCount](#)
- [route53>CreateTrafficPolicyInstance](#)
- [route53>ListHostedZonesByName](#)
- [route53>DeleteTrafficPolicy](#)
- [route53:ChangeTagsForResource](#)
- [route53:UpdateHostedZoneComment](#)
- [route53:GetTrafficPolicyInstanceCount](#)
- [route53:GetReusableDelegationSet](#)
- [route53>ListHealthChecks](#)
- [route53:GetTrafficPolicy](#)
- [route53>ListTagsForResource](#)

Condition context keys for Amazon Route 53

For information about using Route 53 condition keys in an IAM policy, see [Route 53 Keys](#) in the *Amazon Route 53 Developer Guide*.

Amazon Route 53 has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Route 53 Auto Naming

Amazon Route 53 Auto Naming (service prefix: `servicediscovery`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Route 53 Auto Naming

- `servicediscovery:GetOperation`
- `servicediscovery>DeleteService`
- `servicediscovery>DeleteNamespace`
- `servicediscovery>ListServices`
- `servicediscovery>CreatePublicDnsNamespace`
- `servicediscovery>ListNamespaces`
- `servicediscovery>CreatePrivateDnsNamespace`
- `servicediscovery:RegisterInstance`
- `servicediscovery:DeregisterInstance`
- `servicediscoveryGetInstance`
- `servicediscovery>CreateService`
- `servicediscovery:UpdateService`
- `servicediscovery>ListInstances`
- `servicediscovery:GetNamespace`
- `servicediscovery:GetInstanceHealthStatus`
- `servicediscovery:UpdateInstanceHeartbeatStatus`
- `servicediscovery:GetService`
- `servicediscovery>ListOperations`

Condition context keys for Amazon Route 53 Auto Naming

Amazon Route 53 Auto Naming has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Route53 Domains

Amazon Route53 Domains (service prefix: `route53domains`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Route53 Domains

For information about using the following Route 53 API actions in an IAM policy, see [Route 53 Actions](#) in the *Amazon Route 53 Developer Guide*.

- `route53domains:UpdateDomainContactPrivacy`
- `route53domains:TransferDomain`
- `route53domains:DisableDomainTransferLock`

- `route53domains:RenewDomain`
- `route53domains:DisableDomainAutoRenew`
- `route53domains:UpdateDomainNameservers`
- `route53domains:GetDomainSuggestions`
- `route53domains:UpdateDomainContact`
- `route53domains>ListTagsForDomain`
- `route53domains:ResendContactReachabilityEmail`
- `route53domains:UpdateTagsForDomain`
- `route53domains:RetrieveDomainAuthCode`
- `route53domains>DeleteTagsForDomain`
- `route53domains:EnableDomainAutoRenew`
- `route53domains:GetDomainDetail`
- `route53domains>ListDomains`
- `route53domains:ViewBilling`
- `route53domains:CheckDomainAvailability`
- `route53domains:GetContactReachabilityStatus`
- `route53domains>ListOperations`
- `route53domains:GetOperationDetail`
- `route53domains:EnableDomainTransferLock`
- `route53domains:RegisterDomain`

Condition context keys for Amazon Route53 Domains

For information about using Route 53 condition keys in an IAM policy, see [Route 53 Keys](#) in the *Amazon Route 53 Developer Guide*.

Amazon Route53 Domains has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon S3

Amazon S3 (service prefix: s3) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon S3

For information about using the following Amazon S3 API actions in an IAM policy, see [Specifying Permissions in a Policy](#) in the *Amazon Simple Storage Service Developer Guide*.

- `s3:GetBucketLocation`
- `s3:GetBucketCORS`
- `s3:GetIpConfiguration`
- `s3:GetAnalyticsConfiguration`
- `s3:PutObject`
- `s3>ListMultipartUploadParts`
- `s3:DeleteBucketWebsite`
- `s3:DeleteObjectTagging`
- `s3:PutBucketWebsite`
- `s3:GetObjectVersionForReplication`

- s3:GetBucketAcl
- s3:PutLifecycleConfiguration
- s3>DeleteBucket
- s3:GetBucketPolicy
- s3:GetObject
- s3:PutIpConfiguration
- s3:PutObjectTagging
- s3:DeleteObjectVersionTagging
- s3:GetLifecycleConfiguration
- s3:PutReplicationConfiguration
- s3:GetObjectTorrent
- s3:HeadBucket
- s3:GetBucketVersioning
- s3>ListAllMyBuckets
- s3:PutAccelerateConfiguration
- s3:GetReplicationConfiguration
- s3:DeleteObjectVersion
- s3:PutObjectVersionTagging
- s3:GetBucketNotification
- s3:PutMetricsConfiguration
- s3:GetBucketLogging
- s3:PutBucketRequestPayment
- s3:GetInventoryConfiguration
- s3:PutBucketNotification
- s3:GetBucketTagging
- s3>ListBucketByTags
- s3:PutBucketVersioning
- s3:PutBucketCORS
- s3:DeleteObject
- s3:GetObjectVersionTorrent
- s3:ObjectOwnerOverrideToBucketOwner
- s3:ReplicateDelete
- s3:GetObjectVersion
- s3>ListBucketVersions
- s3>ListBucket
- s3:PutBucketTagging
- s3:PutBucketAcl
- s3:PutMetricsConfiguration
- s3:AbortMultipartUpload
- s3:PutBucketPolicy
- s3:PutBucketRequestPayment
- s3:ReplicateObject
- s3:GetObjectAcl
- s3:PutObjectVersionAcl
- s3:GetBucketWebsite

- `s3>ListObjects`
- `s3>ReplicateTags`
- `s3>PutObjectAcl`
- `s3>PutAnalyticsConfiguration`
- `s3>ListBucketMultipartUploads`
- `s3>GetObjectTagging`
- `s3>PutInventoryConfiguration`
- `s3>GetAccelerateConfiguration`
- `s3>RestoreObject`
- `s3>GetObjectVersionTagging`
- `s3>GetObjectVersionAcl`
- `s3>DeleteBucketPolicy`
- `s3>PutBucketLogging`
- `s3>CreateBucket`

Condition context keys for Amazon S3

For information about using the following Amazon S3 conditions in an IAM policy, see [Specifying Conditions in a Policy](#) in the *Amazon Simple Storage Service Developer Guide*.

Amazon S3 has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `s3:ExistingObjectTag/<key>`
- `s3:LocationConstraint`
- `s3:RequestObjectTag/<key>`
- `s3:RequestObjectTagKeys`
- `s3:VersionId`
- `s3:authtype`
- `s3:delimiter`
- `s3:locationconstraint`
- `s3:max-keys`
- `s3:prefix`
- `s3:signatureage`
- `s3:signatureversion`
- `s3:versionid`
- `s3:x-amz-acl`
- `s3:x-amz-content-sha256`
- `s3:x-amz-copy-source`
- `s3:x-amz-grant-full-control`
- `s3:x-amz-grant-read`
- `s3:x-amz-grant-read-acp`
- `s3:x-amz-grant-write`
- `s3:x-amz-grant-write-acp`
- `s3:x-amz-metadata-directive`
- `s3:x-amz-server-side-encryption`

- `s3:x-amz-server-side-encryption-aws-kms-key-id`
- `s3:x-amz-storage-class`
- `s3:x-amz-website-redirect-location`

Actions and Condition Context Keys for Amazon SageMaker

Amazon SageMaker (service prefix: `sagemaker`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon SageMaker

- `sagemaker:StopNotebookInstance`
- `sagemaker>ListModels`
- `sagemaker:DescribeNotebookInstance`
- `sagemaker:StopTrainingJob`
- `sagemaker>CreateEndpointConfig`
- `sagemaker:StartNotebookInstance`
- `sagemaker:DescribeTrainingJob`
- `sagemaker>ListNotebookInstances`
- `sagemaker>DeleteModel`
- `sagemaker>CreateModel`
- `sagemaker:UpdateNotebookInstance`
- `sagemaker>CreatePresignedNotebookInstanceUrl`
- `sagemaker>CreateEndpoint`
- `sagemaker>CreateTrainingJob`
- `sagemaker:DescribeEndpoint`
- `sagemaker:InvokeEndpoint`
- `sagemaker:UpdateEndpointWeightsAndCapacities`
- `sagemaker>ListTrainingJobs`
- `sagemaker>DeleteNotebookInstance`
- `sagemaker:DescribeEndpointConfig`
- `sagemaker>DeleteEndpoint`
- `sagemaker>CreateNotebookInstance`
- `sagemaker>ListEndpoints`
- `sagemaker:UpdateEndpoint`
- `sagemaker>ListEndpointConfigs`
- `sagemaker:DescribeModel`
- `sagemaker:AddTags`
- `sagemaker>ListTags`
- `sagemaker>DeleteEndpointConfig`
- `sagemaker>DeleteTags`

Condition context keys for Amazon SageMaker

Amazon SageMaker has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Security Token Service

AWS Security Token Service (service prefix: sts) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Security Token Service

For information about using the following AWS STS API actions in an IAM policy, see [Granting Permissions to Create Temporary Security Credentials](#) in the *Using Temporary Security Credentials*.

- `sts:GetCallerIdentity`
- `sts:AssumeRoleWithWebIdentity`
- `sts:DecodeAuthorizationMessage`
- `sts:AssumeRole`
- `sts:GetSessionToken`
- `sts:GetFederationToken`
- `sts:AssumeRoleWithSAML`

Condition context keys for AWS Security Token Service

AWS Security Token Service has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `<web-identity-provider>:aud`
- `<web-identity-provider>:oaud`
- `<web-identity-provider>:sub`
- `aws:FederatedProvider`
- `saml:namequalifier`
- `saml:sub`
- `saml:sub_type`
- `saml:aud`
- `saml:iss`
- `saml:doc`

Actions and Condition Context Keys for AWS Service Catalog

AWS Service Catalog (service prefix: servicecatalog) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Service Catalog

For more information about using IAM policies with AWS Service Catalog, see [Authentication and Access Control for AWS Service Catalog](#) in the *AWS Service Catalog Administrator Guide*.

- `servicecatalog>CreatePortfolioShare`
- `servicecatalog:UpdateConstraint`
- `servicecatalog:UpdateProduct`
- `servicecatalog>ListConstraintsForPortfolio`
- `servicecatalog>ListProvisioningArtifacts`

- `servicecatalog:UpdateProvisioningArtifact`
- `servicecatalog>DeleteConstraint`
- `servicecatalog:AssociateProductWithPortfolio`
- `servicecatalog:DescribeProvisioningArtifact`
- `servicecatalog:DescribeRecord`
- `servicecatalog>ListPortfolios`
- `servicecatalog:DescribeProductAsAdmin`
- `servicecatalog>ListLaunchPaths`
- `servicecatalog:DescribePortfolio`
- `servicecatalog:ScanProvisionedProducts`
- `servicecatalog:UpdatePortfolio`
- `servicecatalog>CreatePortfolio`
- `servicecatalog>ListPortfolioAccess`
- `servicecatalog:DescribeProduct`
- `servicecatalog:DeleteProduct`
- `servicecatalog:UpdateProvisionedProduct`
- `servicecatalog:RejectPortfolioShare`
- `servicecatalog>CreateConstraint`
- `servicecatalog>CreateProvisioningArtifact`
- `servicecatalog:DisassociateProductFromPortfolio`
- `servicecatalog>ListAcceptedPortfolioShares`
- `servicecatalog>CreateProduct`
- `servicecatalog:DescribeProvisioningParameters`
- `servicecatalog:TerminateProvisionedProduct`
- `servicecatalog:SearchProductsAsAdmin`
- `servicecatalog>DeletePortfolioShare`
- `servicecatalog>DeletePortfolio`
- `servicecatalog:DescribeConstraint`
- `servicecatalog:AssociatePrincipalWithPortfolio`
- `servicecatalog:AcceptPortfolioShare`
- `servicecatalog>ListRecordHistory`
- `servicecatalog>ListPrincipalsForPortfolio`
- `servicecatalog:SearchProducts`
- `servicecatalog:DisassociatePrincipalFromPortfolio`
- `servicecatalog:DescribeProductView`
- `servicecatalog>ListPortfoliosForProduct`
- `servicecatalog:ProvisionProduct`
- `servicecatalog>DeleteProvisioningArtifact`

Condition context keys for AWS Service Catalog

For example policies that show how these condition keys can be used in an IAM policy, see [Example Access Policies for Provisioned Product Management](#) in the *AWS Service Catalog Administrator Guide*.

AWS Service Catalog has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `servicecatalog:accountLevel`
- `servicecatalog:roleLevel`
- `servicecatalog:userLevel`

Actions and Condition Context Keys for Amazon SES

Amazon SES (service prefix: ses) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon SES

For information about using the following Amazon SES API actions in an IAM policy, see [Controlling User Access to Amazon SES](#) in the *Amazon Simple Email Service Developer Guide*.

- `ses:VerifyEmailIdentity`
- `ses>CreateReceiptRule`
- `ses:UpdateAccountSendingEnabled`
- `ses:SendCustomVerificationEmail`
- `ses>DeleteReceiptFilter`
- `ses>DeleteCustomVerificationEmailTemplate`
- `ses>DeleteIdentityPolicy`
- `ses>CreateReceiptFilter`
- `ses:SetActiveReceiptRuleSet`
- `ses>ListReceiptRuleSets`
- `ses:SendTemplatedEmail`
- `ses:SendRawEmail`
- `ses>CreateCustomVerificationEmailTemplate`
- `ses:UpdateReceiptRule`
- `ses:SendBounce`
- `ses:DescribeActiveReceiptRuleSet`
- `ses:SetIdentityDkimEnabled`
- `ses:PutIdentityPolicy`
- `ses:UpdateConfigurationSetEventDestination`
- `ses:GetIdentityVerificationAttributes`
- `ses>ListVerifiedEmailAddresses`
- `ses:UpdateConfigurationSetTrackingOptions`
- `ses:GetSendStatistics`
- `ses:SetIdentityHeadersInNotificationsEnabled`
- `ses>DeleteReceiptRuleSet`
- `ses>DeleteConfigurationSet`
- `ses:GetTemplate`
- `ses:SetIdentityFeedbackForwardingEnabled`
- `ses>DeleteConfigurationSetEventDestination`
- `ses:GetSendQuota`
- `ses>CreateReceiptRuleSet`
- `ses:SetIdentityNotificationTopic`
- `ses>CreateConfigurationSetTrackingOptions`

- `ses:GetIdentityDkimAttributes`
- `ses:ReorderReceiptRuleSet`
- `ses:VerifyDomainIdentity`
- `ses:TestRenderTemplate`
- `ses:GetIdentityPolicies`
- `ses:DeleteTemplate`
- `ses>CreateTemplate`
- `ses:CreateConfigurationSet`
- `ses>ListCustomVerificationEmailTemplates`
- `ses:SendEmail`
- `ses>ListTemplates`
- `ses:GetCustomVerificationEmailTemplate`
- `ses:VerifyEmailAddress`
- `ses:GetIdentityMailFromDomainAttributes`
- `ses:VerifyDomainDkim`
- `ses:DeleteConfigurationSetTrackingOptions`
- `ses:SendBulkTemplatedEmail`
- `ses:CreateConfigurationSetEventDestination`
- `ses:DeleteReceiptRule`
- `ses:SetReceiptRulePosition`
- `ses:CloneReceiptRuleSet`
- `ses:UpdateTemplate`
- `ses>ListIdentities`
- `ses:UpdateConfigurationSetSendingEnabled`
- `ses:DeleteVerifiedEmailAddress`
- `ses>ListIdentityPolicies`
- `ses:GetIdentityNotificationAttributes`
- `ses:SetIdentityMailFromDomain`
- `ses:GetAccountSendingEnabled`
- `ses:DescribeReceiptRuleSet`
- `ses:UpdateCustomVerificationEmailTemplate`
- `ses:DescribeConfigurationSet`
- `ses>ListConfigurationSets`
- `ses:DeleteIdentity`
- `ses:UpdateConfigurationSetReputationMetricsEnabled`
- `ses>ListReceiptFilters`
- `ses:DescribeReceiptRule`

Condition context keys for Amazon SES

Amazon SES has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `ses:FeedbackAddress`
- `ses:FromAddress`
- `ses:FromDisplayName`

- [ses:Recipients](#)

Actions and Condition Context Keys for AWS Shield

AWS Shield (service prefix: shield) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Shield

For more information about using IAM policies with AWS Shield Advanced and AWS WAF, see [Using Identity-Based Policies \(IAM Policies\) for AWS WAF](#) in the *AWS WAF Developer Guide*.

- [shield:DescribeProtection](#)
- [shield>CreateProtection](#)
- [shield>DeleteProtection](#)
- [shield:DescribeSubscription](#)
- [shield:DescribeAttack](#)
- [shield>DeleteSubscription](#)
- [shield>ListProtections](#)
- [shield>CreateSubscription](#)
- [shield>ListAttacks](#)

Condition context keys for AWS Shield

AWS Shield has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Simple Systems Manager

Amazon Simple Systems Manager (service prefix: ssm) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Simple Systems Manager

For information about using the following Amazon EC2 Systems Manager API actions in an IAM policy, see [Managing Windows Instance Configuration](#) in the *Amazon EC2 User Guide for Windows Instances*.

- [ssm:PutConfigurePackageResult](#)
- [ssm:DescribeMaintenanceWindowExecutions](#)
- [ssm:GetMaintenanceWindowExecutionTaskInvocation](#)
- [ssm:UpdateMaintenanceWindowTarget](#)
- [ssm>ListInstanceAssociations](#) - this is an IAM policy permission only, not an API action that can be called.
- [ssm:StartAutomationExecution](#)
- [ssm:GetInventory](#)
- [ssm:UpdatePatchBaseline](#)
- [ssm:UpdateDocumentDefaultVersion](#)
- [ssm:GetMaintenanceWindowTask](#)
- [ssm:DescribeMaintenanceWindows](#)

- `ssm>ListCommands`
- `ssm>ListAssociationVersions`
- `ssm>UpdateInstanceInformation`
- `ssm>PutComplianceItems`
- `ssm>DescribePatchBaselines`
- `ssm>GetCommandInvocation`
- `ssm>DescribeAssociation`
- `ssm>DescribeParameters`
- `ssm>DescribeMaintenanceWindowTasks`
- `ssm>PutParameter`
- `ssm>UpdateMaintenanceWindow`
- `ssm>AddTagsToResource`
- `ssm>ListInventoryEntries`
- `ssm>StartAssociationsOnce` - this is an IAM policy permission only, not an API action that can be called.
- `ssm>UpdateManagedInstanceRole`
- `ssm>ListTagsForResource`
- `ssm>DeleteDocument`
- `ssm>UpdateAssociation`
- `ssm>RegisterDefaultPatchBaseline`
- `ssm>SendAutomationSignal`
- `ssm>DescribeInstancePatchStates`
- `ssm>DescribePatchGroups`
- `ssm>DeregisterManagedInstance`
- `ssm>DeleteMaintenanceWindow`
- `ssm>ListDocumentVersions`
- `ssm>DeleteResourceDataSync`
- `ssm>DeleteParameters`
- `ssm>GetMaintenanceWindow`
- `ssm>ListResourceDataSync`
- `ssm>GetMaintenanceWindowExecutionTask`
- `ssm>DescribeInstancePatchStatesForPatchGroup`
- `ssm>GetAutomationExecution`
- `ssm>RemoveTagsFromResource`
- `ssm>CreateActivation`
- `ssm>CreateAssociation`
- `ssm>CancelCommand`
- `ssm>ListCommandInvocations`
- `ssm>DeregisterTaskFromMaintenanceWindow`
- `ssm>RegisterPatchBaselineForPatchGroup`
- `ssm>UpdateInstanceAssociationStatus` - this is an IAM policy permission only, not an API action that can be called.
- `ssm>DescribeInstancePatches`
- `ssm>DescribeEffectivePatchesForPatchBaseline`
- `ssm>PutInventory`

- [ssm:DescribeAutomationExecutions](#)
- [ssm>CreateResourceDataSync](#)
- [ssm:GetInventorySchema](#)
- [ssm:GetManifest](#)
- [ssm>DeleteActivation](#)
- [ssm:DescribeAutomationStepExecutions](#)
- [ssm:DescribeMaintenanceWindowExecutionTasks](#)
- [ssm:UpdateAssociationStatus](#)
- [ssm:DescribeMaintenanceWindowExecutionTaskInvocations](#)
- [ssm:DescribeAvailablePatches](#)
- [ssm>ListAssociations](#)
- [ssm:GetParameter](#)
- [ssm:UpdateDocument](#)
- [ssm>DeleteParameter](#)
- [ssm:GetMaintenanceWindowExecution](#)
- [ssm:DescribeActivations](#)
- [ssm:DescribeEffectiveInstanceAssociations](#)
- [ssm:GetDeployablePatchSnapshotForInstance](#)
- [ssm:DescribeInstanceAssociationsStatus](#)
- [ssm:DeregisterTargetFromMaintenanceWindow](#)
- [ssm>CreateMaintenanceWindow](#)
- [ssm:GetParameterHistory](#)
- [ssm:GetParametersByPath](#)
- [ssm:GetPatchBaselineForPatchGroup](#)
- [ssm:DescribeMaintenanceWindowTargets](#)
- [ssm>CreateAssociationBatch](#)
- [ssm:RegisterTaskWithMaintenanceWindow](#)
- [ssm:GetDocument](#)
- [ssm>DeleteAssociation](#)
- [ssm:DescribeDocumentPermission](#)
- [ssm>CreatePatchBaseline](#)
- [ssm:ModifyDocumentPermission](#)
- [ssm:SendCommand](#)
- [ssm:GetDefaultPatchBaseline](#)
- [ssm:DeregisterPatchBaselineForPatchGroup](#)
- [ssm:DescribeInstanceInformation](#)
- [ssm:UpdateMaintenanceWindowTask](#)
- [ssm:RegisterTargetWithMaintenanceWindow](#)
- [ssm>ListDocuments](#)
- [ssm>DeletePatchBaseline](#)
- [ssm:GetPatchBaseline](#)
- [ssm:StopAutomationExecution](#)
- [ssm>CreateDocument](#)
- [ssm:GetParameters](#)
- [ssm:DescribeDocument](#)

- [ssm:DescribePatchGroupState](#)

Condition context keys for Amazon Simple Systems Manager

Amazon Simple Systems Manager has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Simple Workflow Service

Amazon Simple Workflow Service (service prefix: swf) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Simple Workflow Service

For information about using the following Amazon SWF API actions in an IAM policy, see [Using IAM to Manage Access to Amazon SWF Resources](#) in the *Amazon Simple Workflow Service Developer Guide*.

- [swf:StartWorkflowExecution](#)
- [swf:SignalExternalWorkflowExecution](#)
- [swf:ScheduleActivityTask](#)
- [swf:ContinueAsNewWorkflowExecution](#)
- [swf:PollForDecisionTask](#)
- [swf:RespondActivityTaskCanceled](#)
- [swf:RequestCancelActivityTask](#)
- [swf:TerminateWorkflowExecution](#)
- [swf:StartTimer](#)
- [swf:DescribeDomain](#)
- [swf:RespondActivityTaskCompleted](#)
- [swf:RecordMarker](#)
- [swf:RespondDecisionTaskCompleted](#)
- [swf:RecordActivityTaskHeartbeat](#)
- [swf:RequestCancelWorkflowExecution](#)
- [swf:CompleteWorkflowExecution](#)
- [swf:RegisterActivityType](#)
- [swf:CancelTimer](#)
- [swf:CountPendingDecisionTasks](#)
- [swf>ListOpenWorkflowExecutions](#)
- [swf:DescribeWorkflowExecution](#)
- [swf:RequestCancelExternalWorkflowExecution](#)
- [swf>ListActivityTypes](#)
- [swf>ListDomains](#)
- [swf:RegisterDomain](#)
- [swf:CountPendingActivityTasks](#)
- [swf:StartChildWorkflowExecution](#)
- [swf:CancelWorkflowExecution](#)
- [swf:DeprecateActivityType](#)
- [swf:DeprecateWorkflowType](#)

- `swf:DeprecateDomain`
- `swf:PollForActivityTask`
- `swf:DescribeActivityType`
- `swf:FailWorkflowExecution`
- `swf:DescribeWorkflowType`
- `swf:SignalWorkflowExecution`
- `swf:CountClosedWorkflowExecutions`
- `swf:GetWorkflowExecutionHistory`
- `swf:RegisterWorkflowType`
- `swf>ListWorkflowTypes`
- `swf:CountOpenWorkflowExecutions`
- `swf:RespondActivityTaskFailed`
- `swf>ListClosedWorkflowExecutions`

Condition context keys for Amazon Simple Workflow Service

For information about using the following Amazon SWF condition keys in an IAM policy, see [API Summary](#) in the *Amazon Simple Workflow Service Developer Guide*. Each API lists the condition keys that you can use with that API call.

Amazon Simple Workflow Service has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `swf:workflowType.name`
- `swf:activityType.name`
- `swf:activityType.version`
- `swf:defaultTaskList.name`
- `swf:name`
- `swf>tagFilter.tag`
- `swf:tagList.member.0`
- `swf:tagList.member.1`
- `swf:tagList.member.2`
- `swf:tagList.member.3`
- `swf:tagList.member.4`
- `swf:taskList.name`
- `swf:typeFilter.name`
- `swf:typeFilter.version`
- `swf:version`
- `swf:workflowType.name`
- `swf:workflowType.version`

Actions and Condition Context Keys for Amazon SimpleDB

Amazon SimpleDB (service prefix: sdb) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon SimpleDB

For information about using the following Amazon SimpleDB API actions in an IAM policy, see [Amazon SimpleDB Actions](#) in the *Amazon Simple Workflow Service Developer Guide*.

- [sdb:DomainMetadata](#)
- [sdb:GetAttributes](#)
- [sdb:BatchDeleteAttributes](#)
- [sdb:DeleteDomain](#)
- [sdb>CreateDomain](#)
- [sdb:BatchPutAttributes](#)
- [sdb:PutAttributes](#)
- [sdb:Select](#)
- [sdb:DeleteAttributes](#)
- [sdb>ListDomains](#)

Condition context keys for Amazon SimpleDB

For information about using condition keys in an IAM policy to control access to Amazon SimpleDB, see [Amazon SimpleDB Keys](#) in the *Amazon SimpleDB Developer Guide*.

Amazon SimpleDB has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Single Sign-On

Single Sign-On (service prefix: sso) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Single Sign-On

- [sso:UpdateApplicationInstanceSecurityConfiguration](#)
- [sso:PutPermissionsPolicy](#)
- [sso:UpdateProfile](#)
- [sso:UpdateApplicationInstanceResponseConfiguration](#)
- [sso>ListApplicationInstances](#)
- [sso>ListProfileAssociations](#)
- [sso>ListDirectoryAssociations](#)
- [sso:UpdateApplicationInstanceResponseSchemaConfiguration](#)
- [sso>DeleteApplicationInstanceCertificate](#)
- [sso:AssociateDirectory](#)
- [sso:ImportApplicationInstanceServiceProviderMetadata](#)
- [sso:GetTrust](#)
- [sso:DisassociateDirectory](#)
- [sso>DeleteApplicationInstance](#)
- [sso>CreateApplicationInstance](#)
- [sso>CreatePermissionSet](#)
- [sso:UpdateApplicationInstanceServiceProviderConfiguration](#)
- [sso>ListPermissionSets](#)
- [sso:UpdateDirectoryAssociation](#)
- [sso:UpdateApplicationInstanceActiveCertificate](#)

- `sso:UpdateApplicationInstanceStatus`
- `sso:DeleteProfile`
- `sso:DeletePermissionsPolicy`
- `sso:GetApplicationTemplate`
- `sso>ListApplicationTemplates`
- `sso>ListProfiles`
- `sso:DescribePermissionsPolicies`
- `sso:UpdateTrust`
- `sso:DeletePermissionSet`
- `sso:GetApplicationInstance`
- `sso:UpdateApplicationInstanceDisplayData`
- `sso:AssociateProfile`
- `sso>CreateApplicationInstanceCertificate`
- `sso:GetPermissionSet`
- `sso:DisassociateProfile`
- `sso>CreateProfile`
- `sso:StartSSO`
- `sso:GetSSOStatus`
- `sso>ListApplicationInstanceCertificates`
- `sso:CreateTrust`
- `sso:GetProfile`

Condition context keys for Single Sign-On

Single Sign-On has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Snowball

AWS Snowball (service prefix: `snowball`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Snowball

For additional information about using AWS Snowball actions in an IAM policy, see [Authorization and Access Control in Snowball](#) in the *AWS Snowball User Guide*.

- `snowball:DescribeJob`
- `snowball:DescribeAddress`
- `snowball>ListClusterJobs`
- `snowball:DescribeCluster`
- `snowball>CreateAddress`
- `snowball:GetJobManifest`
- `snowball:GetJobUnlockCode`
- `snowball:UpdateJob`
- `snowball:DescribeAddresses`
- `snowball:CancelCluster`
- `snowball:CancelJob`

- `snowball>ListClusters`
- `snowball>CreateJob`
- `snowball>ListJobs`
- `snowball>UpdateCluster`
- `snowball>CreateCluster`
- `snowball>GetSnowballUsage`

Condition context keys for AWS Snowball

AWS Snowball has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon SNS

Amazon SNS (service prefix: sns) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon SNS

For information about using the following Amazon SNS API actions in an IAM policy, see [Amazon SNS Actions](#) in the *Amazon Simple Notification Service Developer Guide*.

- `sns:GetEndpointAttributes`
- `sns>ListTopics`
- `sns:SetSMSAttributes`
- `sns:GetPlatformApplicationAttributes`
- `sns>AddPermission`
- `sns>DeleteTopic`
- `sns>ListPhoneNumbersOptedOut`
- `sns:Subscribe`
- `sns:GetTopicAttributes`
- `sns:GetSubscriptionAttributes`
- `sns:RemovePermission`
- `sns>ListPlatformApplications`
- `sns:SetPlatformApplicationAttributes`
- `sns>CreateTopic`
- `sns>ListSubscriptionsByTopic`
- `sns>ListEndpointsByPlatformApplication`
- `sns:Unsubscribe`
- `sns:SetSubscriptionAttributes`
- `sns:Publish`
- `sns:OptInPhoneNumber`
- `sns:SetEndpointAttributes`
- `sns:GetSMSAttributes`
- `sns>DeleteEndpoint`
- `sns:CheckIfPhoneNumberIsOptedOut`
- `sns:SetTopicAttributes`
- `sns>DeletePlatformApplication`

- `sns:CreatePlatformEndpoint`
- `sns:CreatePlatformApplication`
- `sns>ListSubscriptions`
- `sns:ConfirmSubscription`

Condition context keys for Amazon SNS

For information about using condition keys in an IAM policy to control access to Amazon SNS, see [Amazon SNS Keys](#) in the *Amazon Simple Notification Service Developer Guide*.

Amazon SNS has the following service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

- `sns:Endpoint`
- `sns:Protocol`

Actions and Condition Context Keys for Amazon SQS

Amazon SQS (service prefix: `sq`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon SQS

For information about using the following Amazon SQS API actions in an IAM policy, see [Amazon SQS Actions](#) in the *Amazon Simple Queue Service Developer Guide*.

- `sq:DeleteQueue`
- `sq:PurgeQueue`
- `sq:ChangeMessageVisibilityBatch`
- `sq:SendMessageBatch`
- `sq>ListDeadLetterSourceQueues`
- `sq:AddPermission`
- `sq>ListQueueTags`
- `sq:ChangeMessageVisibility`
- `sq:DeleteMessage`
- `sq:SetQueueAttributes`
- `sq:GetQueueAttributes`
- `sq:ReceiveMessage`
- `sq:DeleteMessageBatch`
- `sq:CreateQueue`
- `sq:RemovePermission`
- `sq:UntagQueue`
- `sq:GetQueueUrl`
- `sq>ListQueues`
- `sq:TagQueue`
- `sq:SendMessage`

Condition context keys for Amazon SQS

For information about using condition keys in an IAM policy to control access to SQS resources, see [Amazon SQS Keys](#) in the *Amazon Simple Queue Service Developer Guide*.

Amazon SQS has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Step Functions

AWS Step Functions (service prefix: states) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Step Functions

AWS Step Functions is capable of executing code and accessing AWS resources (such as data stored in Amazon S3 buckets), so to maintain security, you must grant Step Functions access to those resources. You do this for Step Functions with an IAM role. For information about creating this role, see [Creating IAM Roles for Use with AWS Step Functions](#) in the *AWS Step Functions Developer Guide*.

- `states:DeleteStateMachine`
- `states:DescribeExecution`
- `states:DescribeStateMachineForExecution`
- `states:StopExecution`
- `states:GetExecutionHistory`
- `states>ListExecutions`
- `states:DescribeStateMachine`
- `states:SendTaskSuccess`
- `states:DeleteActivity`
- `states:GetActivityTask`
- `states>ListStateMachines`
- `states:DescribeActivity`
- `states>CreateStateMachine`
- `states:StartExecution`
- `states:SendTaskFailure`
- `states>CreateActivity`
- `states>ListActivities`
- `states:SendTaskHeartbeat`
- `states:UpdateStateMachine`

Condition context keys for AWS Step Functions

AWS Step Functions has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Storage Gateway

Amazon Storage Gateway (service prefix: storagegateway) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Storage Gateway

For information about using the following AWS Storage Gateway API actions in an IAM policy, see [Access Control Using AWS Identity and Access Management \(IAM\)](#) in the *AWS Storage Gateway User Guide*.

Note

For the latest information about permissions in this service, see [AWS Storage Gateway API Permissions: Actions, Resources, and Conditions Reference](#)

- `storagegateway:DescribeGatewayInformation`
- `storagegateway:DescribeUploadBuffer`
- `storagegateway:DescribeCachediSCSIVolumes`
- `storagegateway:UpdateNFSFileShare`
- `storagegateway:DeleteTapeArchive`
- `storagegateway>CreateTapes`
- `storagegateway:UpdateMaintenanceStartTime`
- `storagegateway:DescribeNFSFileShares`
- `storagegateway:DeleteBandwidthRateLimit`
- `storagegateway:CreateCachediSCSIVolume`
- `storagegateway:RetrieveTapeArchive`
- `storagegateway:DeleteVolume`
- `storagegateway:CreateStorediSCSIVolume`
- `storagegateway:DescribeChapCredentials`
- `storagegateway:SetLocalConsolePassword`
- `storagegateway>ListVolumeInitiators`
- `storagegateway:UpdateBandwidthRateLimit`
- `storagegateway:AddCache`
- `storagegateway>ListLocalDisks`
- `storagegateway:StartGateway`
- `storagegateway:ShutdownGateway`
- `storagegateway:DescribeTapeArchives`
- `storagegateway>CreateTapeWithBarcode`
- `storagegateway:UpdateChapCredentials`
- `storagegateway>ListTapes`
- `storagegateway:DescribeBandwidthRateLimit`
- `storagegateway:DeleteFileShare`
- `storagegateway:AddTagsToResource`
- `storagegateway:DisableGateway`
- `storagegateway>CreateSnapshot`
- `storagegateway:DescribeWorkingStorage`
- `storagegateway>ListTagsForResource`
- `storagegateway:ResetCache`
- `storagegateway>CreateSnapshotFromVolumeRecoveryPoint`
- `storagegateway:DescribeTapes`
- `storagegateway:DescribeSnapshotSchedule`
- `storagegateway:DeleteChapCredentials`
- `storagegateway>ListVolumes`
- `storagegateway:DescribeMaintenanceStartTime`
- `storagegateway:UpdateVTLDeviceType`

- `storagegateway>ListVolumeRecoveryPoints`
- `storagegateway>AddWorkingStorage`
- `storagegateway>DescribeVTLDevices`
- `storagegateway>DeleteSnapshotSchedule`
- `storagegateway>RetrieveTapeRecoveryPoint`
- `storagegateway>DeleteGateway`
- `storagegateway>UpdateGatewaySoftwareNow`
- `storagegateway>DescribeCache`
- `storagegateway>ListFileShares`
- `storagegateway>RemoveTagsFromResource`
- `storagegateway>DescribeStorediSCSIVolumes`
- `storagegateway>RefreshCache`
- `storagegateway>ActivateGateway`
- `storagegateway>DeleteTape`
- `storagegateway>CancelArchival`
- `storagegateway>AddUploadBuffer`
- `storagegateway>UpdateGatewayInformation`
- `storagegateway>CancelRetrieval`
- `storagegateway>UpdateSnapshotSchedule`
- `storagegateway>DescribeTapeRecoveryPoints`
- `storagegateway>ListGateways`
- `storagegateway>CreateNFSFileShare`

Condition context keys for Amazon Storage Gateway

Amazon Storage Gateway has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Support

AWS Support (service prefix: support) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Support

AWS Support does not provide service-specific actions or conditions, but your policy must include the `support:*` action (explicitly or implicitly) in order to use the AWS Support Center or to use the AWS Support API. In addition, when you use the AWS Support API to call Trusted Advisor-related actions (such as `DescribeTrustedAdvisorChecks`), none of the `trustedadvisor` actions restrict your access. The `trustedadvisor` actions apply only to Trusted Advisor in the AWS Management Console.

- `support>AddAttachmentsToSet`
- `support>DescribeTrustedAdvisorChecks`
- `support>DescribeAttachment`
- `support>DescribeTrustedAdvisorCheckRefreshStatuses`
- `support>CreateCase`
- `support>DescribeCases`
- `support>DescribeTrustedAdvisorCheckResult`

- support:RefreshTrustedAdvisorCheck
- support:DescribeSeverityLevels
- support:DescribeServices
- support:ResolveCase
- support:AddCommunicationToCase
- support:DescribeTrustedAdvisorCheckSummaries
- support:DescribeCommunications

Condition context keys for AWS Support

AWS Support has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon Translate

Amazon Translate (service prefix: translate) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon Translate

- translate:TranslateText

Condition context keys for Amazon Translate

Amazon Translate has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS Trusted Advisor

AWS Trusted Advisor (service prefix: trustedadvisor) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS Trusted Advisor

AWS Trusted Advisor provides the following service-specific actions and condition context keys for use in IAM policies. Note that these actions apply only to Trusted Advisor in the AWS Management Console; they do not apply to the Trusted Advisor-related actions provided by the AWS Support API (such as `DescribeTrustedAdvisorChecks`). For more information about how these actions affect access to the Trusted Advisor console, see [Controlling Access to the Trusted Advisor Console](#).

To use the Trusted Advisor-related actions provided by the AWS Support API, your policy must include the `support:*` action (explicitly or implicitly); none of the `trustedadvisor` action permissions restrict your access.

- `trustedadvisor:IncludeCheckItems`
- `trustedadvisor:DescribeCheckRefreshStatuses`
- `trustedadvisor:UpdateNotificationPreferences`
- `trustedadvisor:RefreshCheck`
- `trustedadvisor:DescribeNotificationPreferences`
- `trustedadvisor:ExcludeCheckItems`
- `trustedadvisor:DescribeCheckSummaries`

- `trustedadvisor:DescribeCheckItems`

Condition context keys for AWS Trusted Advisor

AWS Trusted Advisor has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS WAF

AWS WAF (service prefix: waf) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS WAF

For information about using the following AWS WAF API actions in an IAM policy, see [Using IAM to Control Access to AWS WAF Resources](#) in the *AWS WAF Developer Guide*.

- `waf:UpdateIPSet`
- `waf:UpdateSizeConstraintSet`
- `waf>CreateGeoMatchSet`
- `waf:GetSampledRequests`
- `waf>DeleteIPSet`
- `waf:GetRateBasedRuleManagedKeys`
- `waf:CreateRateBasedRule`
- `waf>DeleteRule`
- `waf:GetSizeConstraintSet`
- `waf>ListWebACLS`
- `waf>ListSizeConstraintSets`
- `waf>ListRules`
- `waf>DeleteXssMatchSet`
- `waf>CreateWebACL`
- `waf:GetRegexMatchSet`
- `waf:GetIPSet`
- `waf>DeleteRegexMatchSet`
- `waf:GetXssMatchSet`
- `waf:UpdateRule`
- `waf:CreateIPSet`
- `waf:GetByteMatchSet`
- `waf:CreateByteMatchSet`
- `waf:DeleteSizeConstraintSet`
- `waf:UpdateGeoMatchSet`
- `waf>ListIPSets`
- `waf:GetRegexPatternSet`
- `waf:GetSqlInjectionMatchSet`
- `waf>DeleteWebACL`
- `waf:UpdateSqlInjectionMatchSet`
- `waf:DeleteByteMatchSet`
- `waf:GetWebACL`

- [waf>ListXssMatchSets](#)
- [waf>UpdateRegexMatchSet](#)
- [waf>UpdateByteMatchSet](#)
- [waf>CreateXssMatchSet](#)
- [waf>ListByteMatchSets](#)
- [waf>DeleteRegexPatternSet](#)
- [waf>ListRegexPatternSets](#)
- [waf>CreateSqlInjectionMatchSet](#)
- [waf>DeleteGeoMatchSet](#)
- [waf>CreateRule](#)
- [waf>ListRegexMatchSets](#)
- [waf>UpdateRateBasedRule](#)
- [waf>CreateRegexPatternSet](#)
- [waf>UpdateWebACL](#)
- [waf>GetGeoMatchSet](#)
- [waf>ListGeoMatchSets](#)
- [waf>GetRule](#)
- [waf>GetChangeToken](#)
- [waf>ListRateBasedRules](#)
- [waf>GetRateBasedRule](#)
- [waf>DeleteSqlInjectionMatchSet](#)
- [waf>CreateRegexMatchSet](#)
- [waf>UpdateRegexPatternSet](#)
- [waf>GetChangeTokenStatus](#)
- [waf>ListSqlInjectionMatchSets](#)
- [waf>UpdateXssMatchSet](#)
- [waf>DeleteRateBasedRule](#)
- [waf>CreateSizeConstraintSet](#)

Condition context keys for AWS WAF

AWS WAF has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS WAF Regional

AWS WAF Regional (service prefix: waf-regional) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS WAF Regional

For information about using the following AWS WAF API actions in an IAM policy, see [Using IAM to Control Access to AWS WAF Resources](#) in the *AWS WAF Developer Guide*.

- [waf-regional>UpdateIPSet](#)
- [waf-regional>UpdateSizeConstraintSet](#)
- [waf-regional>ListResourcesForWebACL](#)
- [waf-regional>CreateGeoMatchSet](#)

- [waf-regional:DisassociateWebACL](#)
- [waf-regional:GetSampledRequests](#)
- [waf-regional>DeleteIPSet](#)
- [waf-regional:GetRateBasedRuleManagedKeys](#)
- [waf-regional>CreateRateBasedRule](#)
- [waf-regional:GetSizeConstraintSet](#)
- [waf-regional>DeleteRule](#)
- [waf-regional>ListWebACLS](#)
- [waf-regional>ListRules](#)
- [waf-regional>ListSizeConstraintSets](#)
- [waf-regional>DeleteXssMatchSet](#)
- [waf-regional:GetRegexMatchSet](#)
- [waf-regional>CreateWebACL](#)
- [waf-regional:GetIPSet](#)
- [waf-regional>DeleteRegexMatchSet](#)
- [waf-regional:GetXssMatchSet](#)
- [waf-regional:UpdateRule](#)
- [waf-regional>CreateIPSet](#)
- [waf-regional:GetByteMatchSet](#)
- [waf-regional>DeleteSizeConstraintSet](#)
- [waf-regional>CreateByteMatchSet](#)
- [waf-regional>UpdateGeoMatchSet](#)
- [waf-regional>ListIPSets](#)
- [waf-regional:GetRegexPatternSet](#)
- [waf-regional:GetSqlInjectionMatchSet](#)
- [waf-regional>DeleteWebACL](#)
- [waf-regional>UpdateSqlInjectionMatchSet](#)
- [waf-regional>DeleteByteMatchSet](#)
- [waf-regional:GetWebACL](#)
- [waf-regional>ListXssMatchSets](#)
- [waf-regional>UpdateRegexMatchSet](#)
- [waf-regional>CreateXssMatchSet](#)
- [waf-regional>UpdateByteMatchSet](#)
- [waf-regional>ListByteMatchSets](#)
- [waf-regional>DeleteRegexPatternSet](#)
- [waf-regional>ListRegexPatternSets](#)
- [waf-regional>CreateSqlInjectionMatchSet](#)
- [waf-regional>DeleteGeoMatchSet](#)
- [waf-regional>CreateRule](#)
- [waf-regional>GetWebACLForResource](#)
- [waf-regional>ListRegexMatchSets](#)
- [waf-regional>UpdateRateBasedRule](#)
- [waf-regional>CreateRegexPatternSet](#)
- [waf-regional>UpdateWebACL](#)
- [waf-regional>GetGeoMatchSet](#)

- [waf-regional>ListGeoMatchSets](#)
- [waf-regional>GetRule](#)
- [waf-regional>GetChangeToken](#)
- [waf-regional>ListRateBasedRules](#)
- [waf-regional>GetRateBasedRule](#)
- [waf-regional>DeleteSqlInjectionMatchSet](#)
- [waf-regional>CreateRegexMatchSet](#)
- [waf-regional>UpdateRegexPatternSet](#)
- [waf-regional>ListSqlInjectionMatchSets](#)
- [waf-regional>GetChangeTokenStatus](#)
- [waf-regional>AssociateWebACL](#)
- [waf-regional>UpdateXssMatchSet](#)
- [waf-regional>DeleteRateBasedRule](#)
- [waf-regional>CreateSizeConstraintSet](#)

Condition context keys for AWS WAF Regional

AWS WAF Regional has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon WorkDocs

Amazon WorkDocs (service prefix: workdocs) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon WorkDocs

For information about using the following Amazon WorkDocs API actions in an IAM policy, see [IAM Policies for Amazon WorkDocs](#) in the *Amazon WorkDocs Administration Guide*.

- [workdocs>DeleteDocument](#)
- [workdocs>CreateNotificationSubscription](#)
- [workdocs>GetDocumentPath](#)
- [workdocs>DescribeResourcePermissions](#)
- [workdocs>DeleteUser](#)
- [workdocs>CheckAlias](#)
- [workdocs>DescribeFolderContents](#)
- [workdocs>CreateInstance](#)
- [workdocs>GetFolderPath](#)
- [workdocs>GetDocument](#)
- [workdocs>AddUserToGroup](#)
- [workdocs>InitiateDocumentVersionUpload](#)
- [workdocs>DeregisterDirectory](#)
- [workdocs>RemoveAllResourcePermissions](#)
- [workdocs>DeactivateUser](#)
- [workdocs>DeleteFolder](#)
- [workdocs>GetFolder](#)
- [workdocs>UpdateDocumentVersion](#)

- [workdocs:UpdateDocument](#)
- [workdocs:RemoveUserFromGroup](#)
- [workdocs:DescribeUsers](#)
- [workdocs:DescribeDocumentVersions](#)
- [workdocs:RemoveResourcePermission](#)
- [workdocs>DeleteNotificationSubscription](#)
- [workdocs:CreateUser](#)
- [workdocs:DescribeNotificationSubscriptions](#)
- [workdocs:AbortDocumentVersionUpload](#)
- [workdocs:ActivateUser](#)
- [workdocs:UpdateInstanceAlias](#)
- [workdocs:DescribeAvailableDirectories](#)
- [workdocs:AddResourcePermissions](#)
- [workdocs>CreateFolder](#)
- [workdocs:DescribeInstances](#)
- [workdocs:UpdateUser](#)
- [workdocs:DeleteFolderContents](#)
- [workdocs:RegisterDirectory](#)
- [workdocs:GetDocumentVersion](#)
- [workdocs:UpdateFolder](#)
- [workdocs:DeleteInstance](#)

Condition context keys for Amazon WorkDocs

Amazon WorkDocs has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon WorkMail

Amazon WorkMail (service prefix: `workmail`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon WorkMail

For information about using the following Amazon WorkMail API actions in an IAM policy, see [AWS Identity and Access Management Policies for Amazon WorkMail](#) in the *Amazon WorkMail Administrator Guide*.

- [workmail:GetMobileDevicesForUser](#)
- [workmail:GetMobileDeviceDetails](#)
- [workmail>CreateMailDomain](#)
- [workmail:SetMobilePolicyDetails](#)
- [workmail:EnableMailDomain](#)
- [workmail:DescribeDirectories](#)
- [workmail:SearchMembers](#)
- [workmail:DisableMailGroups](#)
- [workmail:DisableMailUsers](#)
- [workmail:AddMembersToGroup](#)

- `workmail:GetMailDomainDetails`
- `workmail>DeleteMailDomain`
- `workmail:ResetUserPassword`
- `workmail:SetAdmin`
- `workmail:SetDefaultMailDomain`
- `workmail:GetMobilePolicyDetails`
- `workmail:GetMailUserDetails`
- `workmail>CreateResource`
- `workmail:DescribeMailUsers`
- `workmail:CreateGroup`
- `workmail:WipeMobileDevice`
- `workmail:GetMailGroupDetails`
- `workmail:SetMailGroupDetails`
- `workmail>ListMembersInMailGroup`
- `workmail:DescribeOrganizations`
- `workmail:CreateOrganization`
- `workmail:DescribeMailDomains`
- `workmail:DescribeMailGroups`
- `workmail:EnableMailGroups`
- `workmail>DeleteMobileDevice`
- `workmail>CreateMailUser`
- `workmail:DescribeKmsKeys`
- `workmail:SetMailUserDetails`
- `workmail:RemoveMembersFromGroup`
- `workmail:EnableMailUsers`
- `workmail>DeleteOrganization`

Condition context keys for Amazon WorkMail

Amazon WorkMail has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for Amazon WorkSpaces

Amazon WorkSpaces (service prefix: `workspaces`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for Amazon WorkSpaces

For information about using the following Amazon WorkSpaces API actions in an IAM policy attached to an IAM user, see [Controlling Access to Amazon WorkSpaces Resources](#) in the *Amazon WorkSpaces Administration Guide*.

- `workspaces:DescribeTags`
- `workspaces:RebootWorkspaces`
- `workspaces:ModifyWorkspaceProperties`
- `workspaces>CreateWorkspaces`
- `workspaces:DescribeWorkspacesConnectionStatus`

- `workspaces:DescribeWorkspaceDirectories`
- `workspaces:TerminateWorkspaces`
- `workspaces:StopWorkspaces`
- `workspaces:DescribeWorkspaces`
- `workspaces:RebuildWorkspaces`
- `workspaces:DescribeWorkspaceBundles`
- `workspaces:DeleteTags`
- `workspaces:CreateTags`
- `workspaces:StartWorkspaces`

Condition context keys for Amazon WorkSpaces

Amazon WorkSpaces has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

Actions and Condition Context Keys for AWS XRay

AWS XRay (service prefix: `xray`) provides the following service-specific actions and condition context keys for use in IAM policies.

Actions for AWS XRay

- `xray:BatchGetTraces`
- `xray:GetServiceGraph`
- `xray:PutTelemetryRecords`
- `xray:PutTraceSegments`
- `xray:GetTraceGraph`
- `xray:GetTraceSummaries`

Condition context keys for AWS XRay

AWS XRay has no service-specific context keys that can be used in an IAM policy. For the list of the global condition context keys that are available to all services, see [Available Global Condition Keys \(p. 477\)](#) in the *IAM Policy Elements Reference*.

IAM Policy Actions Grouped by Access Level

This section lists the access levels that all AWS service actions are grouped into. Access levels categorize AWS service actions based on their behavior. You can use this list to help you understand [policy summaries \(p. 347\)](#) or to determine which actions are relevant to include in your IAM permission policies.

Important

AWS categorizes each AWS action into these access levels based on how closely each action matches the access level definitions in the following list. Before you write any policies that use these access levels, review the definitions and the lists below. Make sure that the actions that you want are categorized the way you expect.

- [List \(p. 616\)](#) — Permission to list resources to determine whether an object exists. Actions with this level of access can list objects but cannot see the contents of a resource. For example, the Amazon S3 action `ListBucket` has the **List** access level.
- [Read \(p. 637\)](#) — Permission to read but not edit the contents and attributes of resources. For example, the Amazon S3 actions `GetObject` and `GetBucketLocation` have the **Read** access level.

- [Write \(p. 663\)](#) — Permission to create, delete, or modify resources. For example, the Amazon S3 actions `CreateBucket`, `DeleteBucket` and `PutObject` have the **Write** access level.
- [Permissions management \(p. 710\)](#) — Permission to grant or modify resource permissions. For example, most IAM and AWS Organizations actions, as well as actions like the Amazon S3 actions `PutBucketPolicy` and `DeleteBucketPolicy` have the **Permissions management** access level.

Tip

To improve the security of your AWS account, restrict or regularly monitor policies that have the **Permissions management** access level classification.

Service Actions Included in the List Access Level

You can use the actions below to list and view details about AWS resources. You cannot use them to read the contents of or modify resources. If an AWS service does not appear on this page then that service does not have any actions in the List category.

Topics

- [Amazon AWS Cloud Contact Center \(p. 618\)](#)
- [Application Discovery \(p. 618\)](#)
- [Amazon Athena \(p. 618\)](#)
- [Auto Scaling \(p. 618\)](#)
- [AWS Batch \(p. 619\)](#)
- [AWS Certificate Manager \(p. 619\)](#)
- [Amazon Cloud Directory \(p. 619\)](#)
- [AWS CloudFormation \(p. 619\)](#)
- [Amazon CloudFront \(p. 619\)](#)
- [AWS CloudHSM \(p. 619\)](#)
- [Amazon CloudSearch \(p. 620\)](#)
- [AWS CloudTrail \(p. 620\)](#)
- [Amazon CloudWatch \(p. 620\)](#)
- [Amazon CloudWatch Events \(p. 620\)](#)
- [Amazon CloudWatch Logs \(p. 620\)](#)
- [AWS CodeBuild \(p. 620\)](#)
- [AWS CodeCommit \(p. 620\)](#)
- [AWS CodeDeploy \(p. 621\)](#)
- [AWS CodePipeline \(p. 621\)](#)
- [AWS CodeStar \(p. 621\)](#)
- [Amazon Cognito Identity \(p. 621\)](#)
- [Amazon Cognito Sync \(p. 621\)](#)
- [Amazon Cognito User Pools \(p. 622\)](#)
- [AWS Config \(p. 622\)](#)
- [Data Pipeline \(p. 622\)](#)
- [AWS Database Migration Service \(p. 622\)](#)
- [AWS Device Farm \(p. 622\)](#)
- [AWS Direct Connect \(p. 623\)](#)
- [AWS Directory Service \(p. 623\)](#)
- [Amazon DynamoDB \(p. 623\)](#)
- [Amazon DynamoDB Accelerator \(DAX\) \(p. 623\)](#)

- [Amazon EC2 \(p. 624\)](#)
- [Amazon EC2 Container Registry \(p. 625\)](#)
- [Amazon EC2 Container Service \(p. 625\)](#)
- [Amazon ElastiCache \(p. 625\)](#)
- [AWS Elastic Beanstalk \(p. 626\)](#)
- [Amazon Elastic File System \(p. 626\)](#)
- [Elastic Load Balancing V2 \(p. 626\)](#)
- [Amazon Elastic MapReduce \(p. 626\)](#)
- [Amazon Elastic Transcoder \(p. 626\)](#)
- [Amazon Elasticsearch Service \(p. 627\)](#)
- [Amazon GameLift \(p. 627\)](#)
- [Amazon Glacier \(p. 627\)](#)
- [Identity And Access Management \(p. 627\)](#)
- [AWS Import Export Disk Service \(p. 628\)](#)
- [Amazon Inspector \(p. 628\)](#)
- [AWS IoT \(p. 628\)](#)
- [AWS Key Management Service \(p. 629\)](#)
- [Amazon Kinesis \(p. 629\)](#)
- [Amazon Kinesis Analytics \(p. 629\)](#)
- [Amazon Kinesis Firehose \(p. 629\)](#)
- [AWS Lambda \(p. 629\)](#)
- [Amazon Lex \(p. 629\)](#)
- [Amazon Lightsail \(p. 630\)](#)
- [Amazon Machine Learning \(p. 630\)](#)
- [AWS Marketplace \(p. 630\)](#)
- [AWS Marketplace Management Portal \(p. 630\)](#)
- [AWS Mobile Hub \(p. 630\)](#)
- [AWS OpsWorks \(p. 631\)](#)
- [AWS OpsWorks Configuration Management \(p. 631\)](#)
- [AWS Organizations \(p. 631\)](#)
- [Amazon Pinpoint \(p. 632\)](#)
- [Amazon Polly \(p. 632\)](#)
- [Amazon RDS \(p. 632\)](#)
- [Amazon Redshift \(p. 633\)](#)
- [Amazon Route 53 \(p. 633\)](#)
- [Amazon Route53 Domains \(p. 633\)](#)
- [Amazon S3 \(p. 633\)](#)
- [Amazon SES \(p. 634\)](#)
- [Amazon SNS \(p. 634\)](#)
- [Amazon SQS \(p. 634\)](#)
- [AWS Service Catalog \(p. 634\)](#)
- [AWS Shield \(p. 634\)](#)
- [Amazon Simple Systems Manager \(p. 635\)](#)
- [Amazon Simple Workflow Service \(p. 635\)](#)
- [Amazon SimpleDB \(p. 635\)](#)
- [AWS Step Functions \(p. 635\)](#)

- [Amazon Storage Gateway \(p. 635\)](#)
- [AWS Trusted Advisor \(p. 635\)](#)
- [AWS WAF \(p. 636\)](#)
- [AWS WAF Regional \(p. 636\)](#)
- [Amazon WorkDocs \(p. 636\)](#)
- [Amazon WorkMail \(p. 636\)](#)
- [Amazon WorkSpaces \(p. 637\)](#)

Amazon AWS Cloud Contact Center

These are the List actions for Amazon AWS Cloud Contact Center.

- `connect>ListInstances`

Application Discovery

These are the List actions for Application Discovery.

- `discovery>ListConfigurations`
- `discovery>ListServerNeighbors`

Amazon Athena

These are the List actions for Amazon Athena.

- `athena>ListNamedQueries`
- `athena>ListQueryExecutions`

Auto Scaling

These are the List actions for Auto Scaling.

- `autoscaling>DescribeAccountLimits`
- `autoscaling>DescribeAdjustmentTypes`
- `autoscaling>DescribeAutoScalingGroups`
- `autoscaling>DescribeAutoScalingInstances`
- `autoscaling>DescribeAutoScalingNotificationTypes`
- `autoscaling>DescribeLaunchConfigurations`
- `autoscaling>DescribeLifecycleHookTypes`
- `autoscaling>DescribeLifecycleHooks`
- `autoscaling>DescribeLoadBalancerTargetGroups`
- `autoscaling>DescribeLoadBalancers`
- `autoscaling>DescribeMetricCollectionTypes`
- `autoscaling>DescribeNotificationConfigurations`
- `autoscaling>DescribePolicies`
- `autoscaling>DescribeScalingActivities`
- `autoscaling>DescribeScalingProcessTypes`
- `autoscaling>DescribeScheduledActions`

- `autoscaling:DescribeTerminationPolicyTypes`

AWS Batch

These are the List actions for AWS Batch.

- `batch>ListJobs`

AWS Certificate Manager

These are the List actions for AWS Certificate Manager.

- `acm>ListCertificates`

Amazon Cloud Directory

These are the List actions for Amazon Cloud Directory.

- `clouddirectory>ListAppliedSchemaArns`
- `clouddirectory>ListDevelopmentSchemaArns`
- `clouddirectory>ListDirectories`
- `clouddirectory>ListPublishedSchemaArns`

AWS CloudFormation

These are the List actions for AWS CloudFormation.

- `cloudformationDescribeStacks`
- `cloudformation>ListChangeSets`
- `cloudformation>ListStackResources`
- `cloudformation>ListStacks`

Amazon CloudFront

These are the List actions for Amazon CloudFront.

- `cloudfront>ListCloudFrontOriginAccessIdentities`
- `cloudfront>ListDistributions`
- `cloudfront>ListDistributionsByWebACLId`
- `cloudfront>ListInvalidations`
- `cloudfront>ListStreamingDistributions`

AWS CloudHSM

These are the List actions for AWS CloudHSM.

- `cloudhsm>ListAvailableZones`
- `cloudhsm>ListHapgs`
- `cloudhsm>ListHsms`
- `cloudhsm>ListLunaClients`

Amazon CloudSearch

These are the List actions for Amazon CloudSearch.

- `cloudsearch:DescribeDomains`
- `cloudsearch>ListDomainNames`

AWS CloudTrail

These are the List actions for AWS CloudTrail.

- `cloudtrail:DescribeTrails`
- `cloudtrail:LookupEvents`

Amazon CloudWatch

These are the List actions for Amazon CloudWatch.

- `cloudwatch:ListMetrics`

Amazon CloudWatch Events

These are the List actions for Amazon CloudWatch Events.

- `events>ListRuleNamesByTarget`
- `events>ListRules`
- `events>ListTargetsByRule`

Amazon CloudWatch Logs

These are the List actions for Amazon CloudWatch Logs.

- `logs:DescribeDestinations`
- `logs:DescribeExportTasks`
- `logs:DescribeLogGroups`
- `logs:DescribeLogStreams`
- `logs:DescribeMetricFilters`
- `logs:DescribeSubscriptionFilters`

AWS CodeBuild

These are the List actions for AWS CodeBuild.

- `codebuild:ListBuilds`
- `codebuild:ListBuildsForProject`
- `codebuild:ListProjects`

AWS CodeCommit

These are the List actions for AWS CodeCommit.

- `codecommit>ListBranches`
- `codecommitListRepositories`

AWS CodeDeploy

These are the List actions for AWS CodeDeploy.

- `codedeployGetApplication`
- `codedeployGetApplicationRevision`
- `codedeployGetDeployment`
- `codedeployGetDeploymentConfig`
- `codedeployGetDeploymentGroup`
- `codedeployGetDeploymentInstance`
- `codedeployGetOnPremisesInstance`
- `codedeployListApplicationRevisions`
- `codedeployListApplications`
- `codedeployListDeploymentConfigs`
- `codedeployListDeploymentGroups`
- `codedeployListDeploymentInstances`
- `codedeployListDeployments`
- `codedeployListOnPremisesInstances`

AWS CodePipeline

These are the List actions for AWS CodePipeline.

- `codepipelineListPipelines`

AWS CodeStar

These are the List actions for AWS CodeStar.

- `codestarListProjects`
- `codestarListResources`
- `codestarListTeamMembers`
- `codestarListUserProfiles`
- `codestarVerifyServiceRole`

Amazon Cognito Identity

These are the List actions for Amazon Cognito Identity.

- `cognito-identityListIdentities`
- `cognito-identityListIdentityPools`

Amazon Cognito Sync

These are the List actions for Amazon Cognito Sync.

- `cognito-sync>ListDatasets`

Amazon Cognito User Pools

These are the List actions for Amazon Cognito User Pools.

- `cognito-idp:AdminListDevices`
- `cognito-idp:AdminListGroupsForUser`
- `cognito-idp>ListDevices`
- `cognito-idp>ListGroups`
- `cognito-idp>ListUserImportJobs`
- `cognito-idp>ListUserPoolClients`
- `cognito-idp>ListUsersInGroup`

AWS Config

These are the List actions for AWS Config.

- `config:DescribeComplianceByConfigRule`
- `config:DescribeComplianceByResource`
- `config:DescribeConfigRuleEvaluationStatus`
- `config:DescribeConfigRules`
- `config:DescribeConfigurationRecorderStatus`
- `config:DescribeConfigurationRecorders`
- `config:DescribeDeliveryChannelStatus`
- `config:DescribeDeliveryChannels`
- `config>ListDiscoveredResources`

Data Pipeline

These are the List actions for Data Pipeline.

- `datapipeline:DescribePipelines`
- `datapipeline:GetAccountLimits`
- `datapipeline>ListPipelines`

AWS Database Migration Service

These are the List actions for AWS Database Migration Service.

- `dms>ListTagsForResource`

AWS Device Farm

These are the List actions for AWS Device Farm.

- `devicefarm>ListArtifacts`
- `devicefarm>ListDevicePools`
- `devicefarm>ListDevices`

- `devicefarm>ListJobs`
- `devicefarm>ListNetworkProfiles`
- `devicefarm>ListOfferingTransactions`
- `devicefarm>ListOfferings`
- `devicefarm>ListProjects`
- `devicefarm>ListRemoteAccessSessions`
- `devicefarm>ListRuns`
- `devicefarm>ListSamples`
- `devicefarm>ListSuites`
- `devicefarm>ListTests`
- `devicefarm>ListUniqueProblems`
- `devicefarm>ListUploads`

AWS Direct Connect

These are the List actions for AWS Direct Connect.

- `directconnect>DescribeConnectionLoa`
- `directconnect>DescribeConnections`
- `directconnect>DescribeConnectionsOnInterconnect`
- `directconnect>DescribeInterconnectLoa`
- `directconnect>DescribeInterconnects`
- `directconnect>DescribeLocations`
- `directconnect>DescribeVirtualGateways`
- `directconnect>DescribeVirtualInterfaces`

AWS Directory Service

These are the List actions for AWS Directory Service.

- `ds>DescribeDirectories`
- `ds>ListSchemaExtensions`

Amazon DynamoDB

These are the List actions for Amazon DynamoDB.

- `dynamodb>ListTables`

Amazon DynamoDB Accelerator (DAX)

These are the List actions for Amazon DynamoDB Accelerator (DAX).

- `dax>DescribeClusters`
- `dax>DescribeDefaultParameters`
- `dax>DescribeEvents`
- `dax>DescribeParameterGroups`
- `dax>DescribeSubnetGroups`
- `dax>ListTables`

Amazon EC2

These are the List actions for Amazon EC2.

- `ec2:DescribeAccountAttributes`
- `ec2:DescribeAddresses`
- `ec2:DescribeAvailabilityZones`
- `ec2:DescribeBundleTasks`
- `ec2:DescribeClassicLinkInstances`
- `ec2:DescribeConversionTasks`
- `ec2:DescribeCustomerGateways`
- `ec2:DescribeDhcpOptions`
- `ec2:DescribeEgressOnlyInternetGateways`
- `ec2:DescribeExportTasks`
- `ec2:DescribeFlowLogs`
- `ec2:DescribeHostReservationOfferings`
- `ec2:DescribeHostReservations`
- `ec2:DescribeHosts`
- `ec2:DescribeIamInstanceProfileAssociation`
- `ec2:DescribeIdFormat`
- `ec2:DescribeIdentityIdFormat`
- `ec2:DescribeImageAttribute`
- `ec2:DescribeImages`
- `ec2:DescribeImportImageTasks`
- `ec2:DescribeImportSnapshotTasks`
- `ec2:DescribeInstanceAttribute`
- `ec2:DescribeInstanceState`
- `ec2:DescribeInstances`
- `ec2:DescribeInternetGateways`
- `ec2:DescribeKeyPairs`
- `ec2:DescribeMovingAddresses`
- `ec2:DescribeNatGateways`
- `ec2:DescribeNetworkAcls`
- `ec2:DescribeNetworkInterfaceAttribute`
- `ec2:DescribeNetworkInterfaces`
- `ec2:DescribePlacementGroups`
- `ec2:DescribePrefixLists`
- `ec2:DescribeRegions`
- `ec2:DescribeReservedInstances`
- `ec2:DescribeReservedInstancesListings`
- `ec2:DescribeReservedInstancesModifications`
- `ec2:DescribeReservedInstancesOfferings`
- `ec2:DescribeRouteTables`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeSnapshotAttribute`
- `ec2:DescribeSnapshots`

- `ec2:DescribeSpotDatafeedSubscription`
- `ec2:DescribeSpotFleetInstances`
- `ec2:DescribeSpotFleetRequestHistory`
- `ec2:DescribeSpotFleetRequests`
- `ec2:DescribeSpotInstanceRequests`
- `ec2:DescribeSpotPriceHistory`
- `ec2:DescribeSubnets`
- `ec2:DescribeVolumeAttribute`
- `ec2:DescribeVolumeStatus`
- `ec2:DescribeVolumes`
- `ec2:DescribeVpcAttribute`
- `ec2:DescribeVpcClassicLink`
- `ec2:DescribeVpcClassicLinkDnsSupport`
- `ec2:DescribeVpcEndpointServices`
- `ec2:DescribeVpcEndpoints`
- `ec2:DescribeVpcPeeringConnections`
- `ec2:DescribeVpcs`
- `ec2:DescribeVpnGateways`

Amazon EC2 Container Registry

These are the List actions for Amazon EC2 Container Registry.

- `ecr:DescribeRepositories`
- `ecr>ListImages`

Amazon EC2 Container Service

These are the List actions for Amazon EC2 Container Service.

- `ecs>ListClusters`
- `ecs>ListContainerInstances`
- `ecs>ListServices`
- `ecs>ListTaskDefinitionFamilies`
- `ecs>ListTaskDefinitions`
- `ecs>ListTasks`

Amazon ElastiCache

These are the List actions for Amazon ElastiCache.

- `elasticache:DescribeCacheClusters`
- `elasticache:DescribeCacheEngineVersions`
- `elasticache:DescribeCacheParameterGroups`
- `elasticache:DescribeCacheParameters`
- `elasticache:DescribeCacheSecurityGroups`
- `elasticache:DescribeCacheSubnetGroups`
- `elasticache:DescribeEngineDefaultParameters`

- elasticache:DescribeEvents
- elasticache:DescribeReplicationGroups
- elasticache:DescribeReservedCacheNodes
- elasticache:DescribeReservedCacheNodesOfferings
- elasticache:DescribeSnapshots
- elasticache>ListAllowedNodeTypeModifications

AWS Elastic Beanstalk

These are the List actions for AWS Elastic Beanstalk.

- elasticbeanstalk:DescribeApplicationVersions
- elasticbeanstalk:DescribeApplications
- elasticbeanstalk:DescribeEnvironments
- elasticbeanstalk>ListAvailableSolutionStacks
- elasticbeanstalk>ListPlatformVersions

Amazon Elastic File System

These are the List actions for Amazon Elastic File System.

- elasticfilesystem:DescribeFileSystems

Elastic Load Balancing V2

These are the List actions for Elastic Load Balancing V2.

- elasticloadbalancing:DescribeListeners
- elasticloadbalancing:DescribeLoadBalancers
- elasticloadbalancing:DescribeTargetGroups
- elasticloadbalancing:DescribeTargetHealth

Amazon Elastic MapReduce

These are the List actions for Amazon Elastic MapReduce.

- elasticmapreduce>ListBootstrapActions
- elasticmapreduce>ListClusters
- elasticmapreduce>ListInstanceGroups
- elasticmapreduce>ListInstances
- elasticmapreduce>ListSecurityConfigurations
- elasticmapreduce>ListSteps
- elasticmapreduce:ViewEventsFromAllClustersInConsole

Amazon Elastic Transcoder

These are the List actions for Amazon Elastic Transcoder.

- elastictranscoder>ListJobsByPipeline

- elastictranscoder>ListJobsByStatus
- elastictranscoder>ListPipelines
- elastictranscoder>ListPresets

Amazon Elasticsearch Service

These are the List actions for Amazon Elasticsearch Service.

- es>DescribeElasticsearchDomain
- es>DescribeElasticsearchDomains
- es>ListDomainNames

Amazon GameLift

These are the List actions for Amazon GameLift.

- gamelift>ListAliases
- gamelift>ListBuilds
- gamelift>ListFleets

Amazon Glacier

These are the List actions for Amazon Glacier.

- glacier>ListJobs
- glacier>ListMultipartUploads
- glacier>ListParts
- glacier>ListProvisionedCapacity
- glacier>ListTagsForVault
- glacier>ListVaults

Identity And Access Management

These are the List actions for Identity And Access Management.

- iam>GetAccountSummary
- iam>GetLoginProfile
- iam>ListAccessKeys
- iam>ListAccountAliases
- iam>ListAttachedGroupPolicies
- iam>ListAttachedRolePolicies
- iam>ListAttachedUserPolicies
- iam>ListEntitiesForPolicy
- iam>ListGroupPolicies
- iam>ListGroups
- iam>ListGroupsForUser
- iam>ListInstanceProfiles
- iam>ListInstanceProfilesForRole
- iam>ListMFADevices

- iam>ListOpenIDConnectProviders
- iam>ListPolicies
- iam>ListPoliciesGrantingServiceAccess
- iam>ListPolicyVersions
- iam>ListRolePolicies
- iam>ListRoles
- iam>ListSAMLProviders
- iam>ListSSHPublicKeys
- iam>ListServerCertificates
- iam>ListServiceSpecificCredentials
- iam>ListSigningCertificates
- iam>ListUserPolicies
- iam>ListUsers
- iam>ListVirtualMFADevices

AWS Import Export Disk Service

These are the List actions for AWS Import Export Disk Service.

- importexport>ListJobs

Amazon Inspector

These are the List actions for Amazon Inspector.

- inspector>ListAssessmentRunAgents
- inspector>ListAssessmentRuns
- inspector>ListAssessmentTargets
- inspector>ListAssessmentTemplates
- inspector>ListEventSubscriptions
- inspector>ListFindings
- inspector>ListRulesPackages
- inspector>ListTagsForResource

AWS IoT

These are the List actions for AWS IoT.

- iot>ListCACertificates
- iot>ListCertificates
- iot>ListCertificatesByCA
- iot>ListOutgoingCertificates
- iot>ListPolicies
- iot>ListPolicyPrincipals
- iot>ListPolicyVersions
- iot>ListPrincipalPolicies
- iot>ListPrincipalThings
- iot>ListThingPrincipals

- `iot>ListThingTypes`
- `iot>ListThings`
- `iot>ListTopicRules`

AWS Key Management Service

These are the List actions for AWS Key Management Service.

- `kms>ListAliases`
- `kms>ListKeyPolicies`
- `kms>ListKeys`
- `kms>ListRetirableGrants`

Amazon Kinesis

These are the List actions for Amazon Kinesis.

- `kinesis>ListStreams`

Amazon Kinesis Analytics

These are the List actions for Amazon Kinesis Analytics.

- `kinesisanalytics>ListApplications`

Amazon Kinesis Firehose

These are the List actions for Amazon Kinesis Firehose.

- `firehose>DescribeDeliveryStream`
- `firehose>ListDeliveryStreams`

AWS Lambda

These are the List actions for AWS Lambda.

- `lambda>ListAliases`
- `lambda>ListEventSourceMappings`
- `lambda>ListFunctions`
- `lambda>ListVersionsByFunction`

Amazon Lex

These are the List actions for Amazon Lex.

- `lex:GetBotAliases`
- `lex:GetBotChannelAssociations`
- `lex:GetBotVersions`
- `lex:GetBots`
- `lex:GetIntentVersions`

- `lex:GetIntents`
- `lex:GetSlotTypeVersions`
- `lex:GetSlotTypes`
- `lex:GetUtterancesView`

Amazon Lightsail

These are the List actions for Amazon Lightsail.

- `lightsail:GetBlueprints`
- `lightsail:GetBundles`
- `lightsail:GetDomain`
- `lightsailGetInstanceSnapshots`
- `lightsail:GetInstanceS`
- `lightsail:GetKeyPair`
- `lightsail:GetRegions`
- `lightsail:GetStaticIps`
- `lightsail:IsVpcPeered`

Amazon Machine Learning

These are the List actions for Amazon Machine Learning.

- `machinelearning:DescribeBatchPredictions`
- `machinelearning:DescribeDataSources`
- `machinelearning:DescribeEvaluations`
- `machinelearning:DescribeMLModels`
- `machinelearning:DescribeTags`

AWS Marketplace

These are the List actions for AWS Marketplace.

- `aws-marketplace:ViewSubscriptions`

AWS Marketplace Management Portal

These are the List actions for AWS Marketplace Management Portal.

- `aws-marketplace-management:viewMarketing`
- `aws-marketplace-management:viewReports`
- `aws-marketplace-management:viewSupport`

AWS Mobile Hub

These are the List actions for AWS Mobile Hub.

- `mobilehub>ListAvailableFeatures`
- `mobilehub>ListAvailableRegions`

- `mobilehub>ListProjects`

AWS OpsWorks

These are the List actions for AWS OpsWorks.

- `opsworks>DescribeAgentVersions`
- `opsworks>DescribeApps`
- `opsworks>DescribeCommands`
- `opsworks>DescribeDeployments`
- `opsworks>DescribeEcsClusters`
- `opsworks>DescribeElasticIps`
- `opsworks>DescribeElasticLoadBalancers`
- `opsworks>DescribeInstances`
- `opsworks>DescribeLayers`
- `opsworks>DescribeLoadBasedAutoScaling`
- `opsworks>DescribeMyUserProfile`
- `opsworks>DescribePermissions`
- `opsworks>DescribeRaidArrays`
- `opsworks>DescribeRdsDbInstances`
- `opsworks>DescribeServiceErrors`
- `opsworks>DescribeStackProvisioningParameters`
- `opsworks>DescribeStackSummary`
- `opsworks>DescribeStacks`
- `opsworks>DescribeTimeBasedAutoScaling`
- `opsworks>DescribeUserProfiles`
- `opsworks>DescribeVolumes`

AWS OpsWorks Configuration Management

These are the List actions for AWS OpsWorks Configuration Management.

- `opsworks-cm>DescribeAccountAttributes`
- `opsworks-cm>DescribeBackups`
- `opsworks-cm>DescribeEvents`
- `opsworks-cm>DescribeNodeAssociationStatus`
- `opsworks-cm>DescribeServers`

AWS Organizations

These are the List actions for AWS Organizations.

- `organizations>ListAccounts`
- `organizations>ListAccountsForParent`
- `organizations>ListChildren`
- `organizations>ListCreateAccountStatus`
- `organizations>ListHandshakesForAccount`
- `organizations>ListHandshakesForOrganization`

- organizations>ListOrganizationalUnitsForParent
- organizations>ListParents
- organizations>ListPolicies
- organizations>ListPoliciesForTarget
- organizations>ListRoots
- organizations>ListTargetsForPolicy

Amazon Pinpoint

These are the List actions for Amazon Pinpoint.

- mobiletargeting>GetApplicationSettings
- mobiletargeting>GetCampaigns
- mobiletargeting>GetImportJobs
- mobiletargeting>GetSegments

Amazon Polly

These are the List actions for Amazon Polly.

- polly>DescribeVoices
- polly>ListLexicons

Amazon RDS

These are the List actions for Amazon RDS.

- rds>DescribeAccountAttributes
- rds>DescribeCertificates
- rds>DescribeDBClusterParameterGroups
- rds>DescribeDBClusterParameters
- rds>DescribeDBClusterSnapshotAttributes
- rds>DescribeDBClusters
- rds>DescribeDBEngineVersions
- rds>DescribeDBInstances
- rds>DescribeDBLogFiles
- rds>DescribeDBParameterGroups
- rds>DescribeDBParameters
- rds>DescribeDBSecurityGroups
- rds>DescribeDBSnapshotAttributes
- rds>DescribeDBSnapshots
- rds>DescribeDBSubnetGroups
- rds>DescribeEngineDefaultClusterParameters
- rds>DescribeEngineDefaultParameters
- rds>DescribeEventCategories
- rds>DescribeEventSubscriptions
- rds>DescribeEvents
- rds>DescribeOptionGroupOptions

- `rds:DescribeOptionGroups`
- `rds:DescribeOrderableDBInstanceOptions`
- `rds:DescribePendingMaintenanceActions`
- `rds:DescribeReservedDBInstances`
- `rds:DescribeReservedDBInstancesOfferings`

Amazon Redshift

These are the List actions for Amazon Redshift.

- `redshift:DescribeClusters`
- `redshift:DescribeEvents`
- `redshift:ViewQueriesInConsole`

Amazon Route 53

These are the List actions for Amazon Route 53.

- `route53:GetChange`
- `route53:GetCheckerIpRanges`
- `route53:GetGeoLocation`
- `route53:GetHealthCheckCount`
- `route53:GetHealthCheckLastFailureReason`
- `route53:GetHealthCheckStatus`
- `route53:GetHostedZone`
- `route53:GetHostedZoneCount`
- `route53:GetReusableDelegationSet`
- `route53>ListGeoLocations`
- `route53>ListHostedZones`
- `route53>ListHostedZonesByName`
- `route53>ListResourceRecordSets`
- `route53>ListReusableDelegationSets`
- `route53>ListTagsForResource`
- `route53>ListTagsForResources`

Amazon Route53 Domains

These are the List actions for Amazon Route53 Domains.

- `route53domains>ListDomains`
- `route53domains>ListOperations`
- `route53domains>ListTagsForDomain`

Amazon S3

These are the List actions for Amazon S3.

- `s3:HeadBucket`
- `s3>ListAllMyBuckets`

- `s3>ListBucket`
- `s3>ListObjects`

Amazon SES

These are the List actions for Amazon SES.

- `ses>ListIdentities`
- `ses>ListIdentityPolicies`
- `ses>ListReceiptFilters`
- `ses>ListReceiptRuleSets`
- `ses>ListVerifiedEmailAddresses`

Amazon SNS

These are the List actions for Amazon SNS.

- `sns>ListEndpointsByPlatformApplication`
- `sns>ListPlatformApplications`
- `sns>ListSubscriptions`
- `sns>ListSubscriptionsByTopic`
- `sns>ListTopics`

Amazon SQS

These are the List actions for Amazon SQS.

- `sqs>ListQueues`

AWS Service Catalog

These are the List actions for AWS Service Catalog.

- `servicecatalog>ListAcceptedPortfolioShares`
- `servicecatalog>ListConstraintsForPortfolio`
- `servicecatalog>ListLaunchPaths`
- `servicecatalog>ListPortfolioAccess`
- `servicecatalog>ListPortfolios`
- `servicecatalog>ListPortfoliosForProduct`
- `servicecatalog>ListPrincipalsForPortfolio`
- `servicecatalog>ListProvisioningArtifacts`
- `servicecatalog>ListRecordHistory`
- `servicecatalog>ScanProvisionedProducts`
- `servicecatalog>SearchProducts`
- `servicecatalog>SearchProductsAsAdmin`

AWS Shield

These are the List actions for AWS Shield.

- `shield>ListAttacks`
- `shield>ListProtections`

Amazon Simple Systems Manager

These are the List actions for Amazon Simple Systems Manager.

- `ssm>ListAssociations`
- `ssm>ListDocumentVersions`
- `ssm>ListDocuments`
- `ssm>ListInstanceAssociations`
- `ssm>ListInventoryEntries`

Amazon Simple Workflow Service

These are the List actions for Amazon Simple Workflow Service.

- `swf>ListActivityTypes`
- `swf>ListClosedWorkflowExecutions`
- `swf>ListDomains`
- `swf>ListOpenWorkflowExecutions`
- `swf>ListWorkflowTypes`

Amazon SimpleDB

These are the List actions for Amazon SimpleDB.

- `sdb>ListDomains`

AWS Step Functions

These are the List actions for AWS Step Functions.

- `states>ListActivities`
- `states>ListStateMachine`

Amazon Storage Gateway

These are the List actions for Amazon Storage Gateway.

- `storagegateway>ListGateways`
- `storagegateway>ListLocalDisks`
- `storagegateway>ListVolumeRecoveryPoints`
- `storagegateway>ListVolumes`

AWS Trusted Advisor

These are the List actions for AWS Trusted Advisor.

- `trustedadvisor>DescribeCheckItems`

- `trustedadvisor:DescribeCheckRefreshStatuses`
- `trustedadvisor:DescribeCheckSummaries`
- `trustedadvisor:DescribeNotificationPreferences`

AWS WAF

These are the List actions for AWS WAF.

- `waf>ListByteMatchSets`
- `waf>ListIPSets`
- `waf>ListRateBasedules`
- `waf>ListRules`
- `waf>ListSizeConstraintSets`
- `waf>ListSqlInjectionMatchSets`
- `waf>ListWebACLS`
- `waf>ListXssMatchSets`

AWS WAF Regional

These are the List actions for AWS WAF Regional.

- `waf-regional>ListByteMatchSets`
- `waf-regional>ListIPSets`
- `waf-regional>ListRateBasedules`
- `waf-regional>ListResourcesForWebACL`
- `waf-regional>ListRules`
- `waf-regional>ListSizeConstraintSets`
- `waf-regional>ListSqlInjectionMatchSets`
- `waf-regional>ListWebACLS`
- `waf-regional>ListXssMatchSets`

Amazon WorkDocs

These are the List actions for Amazon WorkDocs.

- `workdocs:DescribeAvailableDirectories`
- `workdocs:DescribeDocumentVersions`
- `workdocs:DescribeFolderContents`
- `workdocs:DescribeInstances`
- `workdocs:DescribeNotificationSubscriptions`
- `workdocs:DescribeResourcePermissions`
- `workdocs:DescribeUsers`

Amazon WorkMail

These are the List actions for Amazon WorkMail.

- `workmail:DescribeDirectories`

- `workmail:DescribeKmsKeys`
- `workmail:DescribeMailDomains`
- `workmail:DescribeMailGroups`
- `workmail:DescribeMailUsers`
- `workmail:DescribeOrganizations`

Amazon WorkSpaces

These are the List actions for Amazon WorkSpaces.

- `workspaces:DescribeTags`
- `workspaces:DescribeWorkspaceBundles`
- `workspaces:DescribeWorkspaceDirectories`
- `workspaces:DescribeWorkspaces`

Service Actions Included in the Read Access Level

You can use the actions below to read content in AWS resources. If an AWS service does not appear on this page then that service does not have any actions in the Read category.

Topics

- [Amazon AWS Cloud Contact Center \(p. 639\)](#)
- [Amazon AppStream \(p. 639\)](#)
- [Application Auto Scaling \(p. 639\)](#)
- [Application Discovery \(p. 640\)](#)
- [Amazon Athena \(p. 640\)](#)
- [Auto Scaling \(p. 640\)](#)
- [AWS Batch \(p. 640\)](#)
- [AWS Billing \(p. 640\)](#)
- [AWS Budget Service \(p. 640\)](#)
- [AWS Certificate Manager \(p. 641\)](#)
- [Amazon Cloud Directory \(p. 641\)](#)
- [AWS CloudFormation \(p. 641\)](#)
- [Amazon CloudFront \(p. 642\)](#)
- [AWS CloudHSM \(p. 642\)](#)
- [Amazon CloudSearch \(p. 642\)](#)
- [AWS CloudTrail \(p. 642\)](#)
- [Amazon CloudWatch \(p. 643\)](#)
- [Amazon CloudWatch Events \(p. 643\)](#)
- [Amazon CloudWatch Logs \(p. 643\)](#)
- [AWS CodeBuild \(p. 643\)](#)
- [AWS CodeCommit \(p. 643\)](#)
- [AWS CodeDeploy \(p. 644\)](#)
- [AWS CodePipeline \(p. 644\)](#)
- [AWS CodeStar \(p. 644\)](#)
- [Amazon Cognito Identity \(p. 644\)](#)
- [Amazon Cognito Sync \(p. 644\)](#)

- [Amazon Cognito User Pools \(p. 645\)](#)
- [AWS Config \(p. 645\)](#)
- [AWS Cost and Usage Report \(p. 645\)](#)
- [Data Pipeline \(p. 645\)](#)
- [AWS Database Migration Service \(p. 646\)](#)
- [AWS Device Farm \(p. 646\)](#)
- [AWS Direct Connect \(p. 646\)](#)
- [AWS Directory Service \(p. 647\)](#)
- [Amazon DynamoDB \(p. 647\)](#)
- [Amazon DynamoDB Accelerator \(DAX\) \(p. 647\)](#)
- [Amazon EC2 \(p. 648\)](#)
- [Amazon EC2 Container Registry \(p. 648\)](#)
- [Amazon EC2 Container Service \(p. 648\)](#)
- [Amazon ElastiCache \(p. 648\)](#)
- [AWS Elastic Beanstalk \(p. 648\)](#)
- [Amazon Elastic File System \(p. 649\)](#)
- [Elastic Load Balancing V2 \(p. 649\)](#)
- [Amazon Elastic MapReduce \(p. 649\)](#)
- [Amazon Elastic Transcoder \(p. 649\)](#)
- [Amazon Elasticsearch Service \(p. 650\)](#)
- [Amazon GameLift \(p. 650\)](#)
- [Amazon Glacier \(p. 650\)](#)
- [AWS Health APIs and Notifications \(p. 650\)](#)
- [Identity And Access Management \(p. 651\)](#)
- [AWS Import Export Disk Service \(p. 651\)](#)
- [Amazon Inspector \(p. 651\)](#)
- [AWS IoT \(p. 652\)](#)
- [AWS Key Management Service \(p. 652\)](#)
- [Amazon Kinesis \(p. 652\)](#)
- [Amazon Kinesis Analytics \(p. 653\)](#)
- [AWS Lambda \(p. 653\)](#)
- [Amazon Lex \(p. 653\)](#)
- [Amazon Lightsail \(p. 653\)](#)
- [Amazon Machine Learning \(p. 654\)](#)
- [Manage Amazon API Gateway \(p. 654\)](#)
- [Amazon Mechanical Turk \(p. 654\)](#)
- [Amazon Message Delivery Service \(p. 654\)](#)
- [Amazon Mobile Analytics \(p. 655\)](#)
- [AWS Mobile Hub \(p. 655\)](#)
- [AWS OpsWorks \(p. 655\)](#)
- [AWS Organizations \(p. 655\)](#)
- [Amazon Pinpoint \(p. 655\)](#)
- [Amazon Polly \(p. 656\)](#)
- [Amazon RDS \(p. 656\)](#)
- [Amazon Redshift \(p. 656\)](#)

- [Amazon Rekognition \(p. 656\)](#)
- [Amazon Resource Group Tagging API \(p. 657\)](#)
- [Amazon Route 53 \(p. 657\)](#)
- [Amazon Route53 Domains \(p. 657\)](#)
- [Amazon S3 \(p. 657\)](#)
- [Amazon SES \(p. 658\)](#)
- [Amazon SNS \(p. 659\)](#)
- [Amazon SQS \(p. 659\)](#)
- [AWS Security Token Service \(p. 659\)](#)
- [AWS Service Catalog \(p. 659\)](#)
- [AWS Shield \(p. 659\)](#)
- [Amazon Simple Systems Manager \(p. 660\)](#)
- [Amazon Simple Workflow Service \(p. 661\)](#)
- [Amazon SimpleDB \(p. 661\)](#)
- [AWS Step Functions \(p. 661\)](#)
- [Amazon Storage Gateway \(p. 661\)](#)
- [AWS WAF \(p. 662\)](#)
- [AWS WAF Regional \(p. 662\)](#)
- [Amazon WorkDocs \(p. 662\)](#)
- [Amazon WorkMail \(p. 663\)](#)
- [Amazon WorkSpaces \(p. 663\)](#)
- [AWS XRay \(p. 663\)](#)

Amazon AWS Cloud Contact Center

These are the Read actions for Amazon AWS Cloud Contact Center.

- `connect:DescribeInstance`
- `connect:GetFederationTokens`

Amazon AppStream

These are the Read actions for Amazon AppStream.

- `appstream:DescribeFleets`
- `appstream:DescribeImages`
- `appstream:DescribeSessions`
- `appstream:DescribeStacks`
- `appstream>ListAssociatedFleets`
- `appstream>ListAssociatedStacks`

Application Auto Scaling

These are the Read actions for Application Auto Scaling.

- `application-autoscaling:DescribeScalableTargets`
- `application-autoscaling:DescribeScalingActivities`

- `application-autoscaling:DescribeScalingPolicies`

Application Discovery

These are the Read actions for Application Discovery.

- `discovery:DescribeAgents`
- `discovery:DescribeConfigurations`
- `discovery:DescribeExportConfigurations`
- `discovery:DescribeTags`
- `discovery:GetDiscoverySummary`

Amazon Athena

These are the Read actions for Amazon Athena.

- `athena:BatchGetNamedQuery`
- `athena:BatchGetQueryExecution`
- `athena:GetNamedQuery`
- `athena:GetQueryExecution`
- `athena:GetQueryResults`

Auto Scaling

These are the Read actions for Auto Scaling.

- `autoscaling:DescribeTags`

AWS Batch

These are the Read actions for AWS Batch.

- `batch:DescribeComputeEnvironments`
- `batch:DescribeJobDefinitions`
- `batch:DescribeJobQueues`
- `batch:DescribeJobs`

AWS Billing

These are the Read actions for AWS Billing.

- `aws-portal:ViewAccount`
- `aws-portal:ViewBilling`
- `aws-portal:ViewPaymentMethods`
- `aws-portal:ViewUsage`

AWS Budget Service

These are the Read actions for AWS Budget Service.

- budgets:ViewBudget

AWS Certificate Manager

These are the Read actions for AWS Certificate Manager.

- acm:DescribeCertificate
- acm:GetCertificate
- acm>ListTagsForCertificate

Amazon Cloud Directory

These are the Read actions for Amazon Cloud Directory.

- clouddirectory:BatchRead
- clouddirectory:GetDirectory
- clouddirectory:GetFacet
- clouddirectory:GetObjectInformation
- clouddirectory:GetSchemaAsJson
- clouddirectory:GetTypedLinkFacetInformation
- clouddirectory>ListAttachedIndices
- clouddirectory>ListFacetAttributes
- clouddirectory>ListFacetNames
- clouddirectory>ListIncomingTypedLinks
- clouddirectory>ListIndex
- clouddirectory>ListObjectAttributes
- clouddirectory>ListObjectChildren
- clouddirectory>ListObjectParentPaths
- clouddirectory>ListObjectParents
- clouddirectory>ListObjectPolicies
- clouddirectory>ListOutgoingTypedLinks
- clouddirectory>ListPolicyAttachments
- clouddirectory>ListTagsForResource
- clouddirectory>ListTypedLinkFacetAttributes
- clouddirectory>ListTypedLinkFacetNames
- clouddirectory:LookupPolicy

AWS CloudFormation

These are the Read actions for AWS CloudFormation.

- cloudformation:DescribeAccountLimits
- cloudformation:DescribeChangeSet
- cloudformation:DescribeStackEvents
- cloudformation:DescribeStackResource
- cloudformation:DescribeStackResources
- cloudformation:EstimateTemplateCost
- cloudformation:GetStackPolicy

- `cloudformation:GetTemplate`
- `cloudformation:GetTemplateSummary`
- `cloudformation:PreviewStackUpdate`

Amazon CloudFront

These are the Read actions for Amazon CloudFront.

- `cloudfront:GetCloudFrontOriginAccessIdentity`
- `cloudfront:GetCloudFrontOriginAccessIdentityConfig`
- `cloudfront:GetDistribution`
- `cloudfront:GetDistributionConfig`
- `cloudfront:GetInvalidation`
- `cloudfront:GetStreamingDistribution`
- `cloudfront:GetStreamingDistributionConfig`
- `cloudfront>ListTagsForResource`

AWS CloudHSM

These are the Read actions for AWS CloudHSM.

- `cloudhsm:DescribeHapg`
- `cloudhsm:DescribeHsm`
- `cloudhsm:DescribeLunaClient`
- `cloudhsm:GetConfig`
- `cloudhsm>ListTagsForResource`

Amazon CloudSearch

These are the Read actions for Amazon CloudSearch.

- `cloudsearch:DescribeAnalysisSchemes`
- `cloudsearch:DescribeAvailabilityOptions`
- `cloudsearch:DescribeExpressions`
- `cloudsearch:DescribeIndexFields`
- `cloudsearch:DescribeScalingParameters`
- `cloudsearch:DescribeServiceAccessPolicies`
- `cloudsearch:DescribeSuggesters`
- `cloudsearch>ListTags`
- `cloudsearch:search`
- `cloudsearch:suggest`

AWS CloudTrail

These are the Read actions for AWS CloudTrail.

- `cloudtrail:GetEventSelectors`
- `cloudtrail:GetTrailStatus`

- `cloudtrail>ListPublicKeys`
- `cloudtrail>ListTags`

Amazon CloudWatch

These are the Read actions for Amazon CloudWatch.

- `cloudwatch>DescribeAlarmHistory`
- `cloudwatch>DescribeAlarms`
- `cloudwatch>DescribeAlarmsForMetric`
- `cloudwatch>GetMetricData`
- `cloudwatch>GetMetricStatistics`

Amazon CloudWatch Events

These are the Read actions for Amazon CloudWatch Events.

- `events>DescribeRule`
- `events>TestEventPattern`

Amazon CloudWatch Logs

These are the Read actions for Amazon CloudWatch Logs.

- `logs>FilterLogEvents`
- `logs>GetLogEvents`
- `logs>TestMetricFilter`

AWS CodeBuild

These are the Read actions for AWS CodeBuild.

- `codebuild>BatchGetBuilds`
- `codebuild>BatchGetProjects`
- `codebuild>ListConnectedOAuthAccounts`
- `codebuild>ListCuratedEnvironmentImages`
- `codebuild>ListRepositories`

AWS CodeCommit

These are the Read actions for AWS CodeCommit.

- `codecommit>BatchGetRepositories`
- `codecommit>GetBlob`
- `codecommit>GetBranch`
- `codecommit>GetCommit`
- `codecommit>GetObjectIdentifier`
- `codecommit>GetReferences`
- `codecommit>GetRepository`

- `codecommit:GetRepositoryTriggers`
- `codecommit:GetTree`
- `codecommit:GitPull`

AWS CodeDeploy

These are the Read actions for AWS CodeDeploy.

- `codedeploy:BatchGetApplicationRevisions`
- `codedeploy:BatchGetApplications`
- `codedeploy:BatchGetDeploymentGroups`
- `codedeploy:BatchGetDeploymentInstances`
- `codedeploy:BatchGetDeployments`
- `codedeploy:BatchGetOnPremisesInstances`

AWS CodePipeline

These are the Read actions for AWS CodePipeline.

- `codepipeline:GetJobDetails`
- `codepipeline:GetPipeline`
- `codepipeline:GetPipelineExecution`
- `codepipeline:GetPipelineState`
- `codepipeline:GetThirdPartyJobDetails`
- `codepipeline>ListActionTypes`

AWS CodeStar

These are the Read actions for AWS CodeStar.

- `codestar:DescribeProject`
- `codestar:DescribeUserProfile`
- `codestar:GetExtendedAccess`

Amazon Cognito Identity

These are the Read actions for Amazon Cognito Identity.

- `cognito-identity:DescribeIdentity`
- `cognito-identity:DescribeIdentityPool`
- `cognito-identity:GetCredentialsForIdentity`
- `cognito-identity:GetIdentityPoolRoles`
- `cognito-identity:GetOpenIdToken`
- `cognito-identity:GetOpenIdTokenForDeveloperIdentity`
- `cognito-identity:LookupDeveloperIdentity`

Amazon Cognito Sync

These are the Read actions for Amazon Cognito Sync.

- `cognito-sync:DescribeDataset`
- `cognito-sync:DescribeIdentityPoolUsage`
- `cognito-sync:DescribeIdentityUsage`
- `cognito-sync:GetBulkPublishDetails`
- `cognito-sync:GetCognitoEvents`
- `cognito-sync:GetIdentityPoolConfiguration`
- `cognito-sync>ListIdentityPoolUsage`
- `cognito-sync>ListRecords`
- `cognito-sync:QueryRecords`

Amazon Cognito User Pools

These are the Read actions for Amazon Cognito User Pools.

- `cognito-idp:AdminGetDevice`
- `cognito-idp:Admin GetUser`
- `cognito-idp:DescribeUserImportJob`
- `cognito-idp:DescribeUserPool`
- `cognito-idp:DescribeUserPoolClient`
- `cognito-idp:GetCSVHeader`
- `cognito-idp:GetDevice`
- `cognito-idp:GetGroup`
- `cognito-idp: GetUser`
- `cognito-idp: GetUserAttributeVerificationCode`

AWS Config

These are the Read actions for AWS Config.

- `config:DeliverConfigSnapshot`
- `config:GetComplianceDetailsByConfigRule`
- `config:GetComplianceDetailsByResource`
- `config:GetComplianceSummaryByConfigRule`
- `config:GetComplianceSummaryByResourceType`
- `config:GetResourceConfigHistory`
- `config:GetResources`
- `config:GetTagKeys`

AWS Cost and Usage Report

These are the Read actions for AWS Cost and Usage Report.

- `cur:DescribeReportDefinitions`

Data Pipeline

These are the Read actions for Data Pipeline.

- `datapipeline:DescribeObjects`

- `datapipeline:EvaluateExpression`
- `datapipeline:GetPipelineDefinition`
- `datapipeline:QueryObjects`
- `datapipeline:ValidatePipelineDefinition`

AWS Database Migration Service

These are the Read actions for AWS Database Migration Service.

- `dms:DescribeAccountAttributes`
- `dms:DescribeCertificates`
- `dms:DescribeConnections`
- `dms:DescribeEndpointTypes`
- `dms:DescribeEndpoints`
- `dms:DescribeEventCategories`
- `dms:DescribeEventSubscriptions`
- `dms:DescribeEvents`
- `dms:DescribeOrderableReplicationInstances`
- `dms:DescribeRefreshSchemasStatus`
- `dms:DescribeReplicationInstances`
- `dms:DescribeReplicationSubnetGroups`
- `dms:DescribeReplicationTasks`
- `dms:DescribeSchemas`
- `dms:DescribeTableStatistics`
- `dms:TestConnection`

AWS Device Farm

These are the Read actions for AWS Device Farm.

- `devicefarm:GetAccountSettings`
- `devicefarm:GetDevice`
- `devicefarm:GetDevicePool`
- `devicefarm:GetDevicePoolCompatibility`
- `devicefarm:GetJob`
- `devicefarm:GetNetworkProfile`
- `devicefarm:GetOfferingStatus`
- `devicefarm:GetProject`
- `devicefarm:GetRemoteAccessSession`
- `devicefarm:GetRun`
- `devicefarm:GetSuite`
- `devicefarm:GetTest`
- `devicefarm:GetUpload`

AWS Direct Connect

These are the Read actions for AWS Direct Connect.

- `directconnect:ConfirmConnection`
- `directconnect:ConfirmPrivateVirtualInterface`
- `directconnect:ConfirmPublicVirtualInterface`

AWS Directory Service

These are the Read actions for AWS Directory Service.

- `ds:DescribeConditionalForwarders`
- `ds:DescribeEventTopics`
- `ds:DescribeSnapshots`
- `ds:DescribeTrusts`
- `ds:GetDirectoryLimits`
- `ds:GetSnapshotLimits`
- `ds>ListAuthorizedApplications`
- `ds>ListIpRoutes`
- `ds>ListTagsForResource`
- `ds:VerifyTrust`

Amazon DynamoDB

These are the Read actions for Amazon DynamoDB.

- `dynamodb:BatchGetItem`
- `dynamodb:DescribeLimits`
- `dynamodb:DescribeReservedCapacity`
- `dynamodb:DescribeReservedCapacityOfferings`
- `dynamodb:DescribeStream`
- `dynamodb:DescribeTable`
- `dynamodb:DescribeTimeToLive`
- `dynamodb:.GetItem`
- `dynamodb:GetRecords`
- `dynamodb:GetShardIterator`
- `dynamodb>ListStreams`
- `dynamodb>ListTagsOfResource`
- `dynamodb:Query`
- `dynamodb:Scan`

Amazon DynamoDB Accelerator (DAX)

These are the Read actions for Amazon DynamoDB Accelerator (DAX).

- `dax:BatchGetItem`
- `dax:DescribeParameters`
- `dax:DescribeTable`
- `dax:.GetItem`
- `dax>ListTags`
- `dax:Query`

- `dax:Scan`

Amazon EC2

These are the Read actions for Amazon EC2.

- `ec2:DescribeScheduledInstanceAvailability`
- `ec2:DescribeScheduledInstances`
- `ec2:DescribeSecurityGroupReferences`
- `ec2:DescribeStaleSecurityGroups`
- `ec2:DescribeTags`
- `ec2:DescribeVolumesModifications`
- `ec2:DescribeVpnConnections`
- `ec2:GetConsoleOutput`
- `ec2:GetConsoleScreenshot`
- `ec2:GetHostReservationPurchasePreview`
- `ec2:GetPasswordData`
- `ec2:GetReservedInstancesExchangeQuote`

Amazon EC2 Container Registry

These are the Read actions for Amazon EC2 Container Registry.

- `ecr:BatchCheckLayerAvailability`
- `ecr:BatchGetImage`
- `ecr:DescribeImages`
- `ecr:GetAuthorizationToken`
- `ecr:GetDownloadUrlForLayer`
- `ecr:GetRepositoryPolicy`

Amazon EC2 Container Service

These are the Read actions for Amazon EC2 Container Service.

- `ecs:DescribeClusters`
- `ecs:DescribeContainerInstances`
- `ecs:DescribeServices`
- `ecs:DescribeTaskDefinition`
- `ecs:DescribeTasks`

Amazon ElastiCache

These are the Read actions for Amazon ElastiCache.

- `elasticache>ListTagsForResource`

AWS Elastic Beanstalk

These are the Read actions for AWS Elastic Beanstalk.

- elasticbeanstalk:CheckDNSAvailability
- elasticbeanstalk:DescribeConfigurationOptions
- elasticbeanstalk:DescribeConfigurationSettings
- elasticbeanstalk:DescribeEnvironmentHealth
- elasticbeanstalk:DescribeEnvironmentManagedActionHistory
- elasticbeanstalk:DescribeEnvironmentManagedActions
- elasticbeanstalk:DescribeEnvironmentResources
- elasticbeanstalk:DescribeEvents
- elasticbeanstalk:DescribeInstancesHealth
- elasticbeanstalk:DescribePlatformVersion
- elasticbeanstalk:RequestEnvironmentInfo
- elasticbeanstalk:RetrieveEnvironmentInfo
- elasticbeanstalk:ValidateConfigurationSettings

Amazon Elastic File System

These are the Read actions for Amazon Elastic File System.

- elasticfilesystem:DescribeMountTargetSecurityGroups
- elasticfilesystem:DescribeMountTargets
- elasticfilesystem:DescribeTags

Elastic Load Balancing V2

These are the Read actions for Elastic Load Balancing V2.

- elasticloadbalancing:DescribeLoadBalancerAttributes
- elasticloadbalancing:DescribeRules
- elasticloadbalancing:DescribeSSLPolicies
- elasticloadbalancing:DescribeTags
- elasticloadbalancing:DescribeTargetGroupAttributes

Amazon Elastic MapReduce

These are the Read actions for Amazon Elastic MapReduce.

- elasticmapreduce:DescribeCluster
- elasticmapreduce:DescribeJobFlows
- elasticmapreduce:DescribeSecurityConfiguration
- elasticmapreduce:DescribeStep

Amazon Elastic Transcoder

These are the Read actions for Amazon Elastic Transcoder.

- elastictranscoder:ReadJob
- elastictranscoder:ReadPipeline
- elastictranscoder:ReadPreset

Amazon Elasticsearch Service

These are the Read actions for Amazon Elasticsearch Service.

- `es:DescribeElasticsearchDomainConfig`
- `es:ESHttpGet`
- `es:ESHttpGetHead`
- `es>ListTags`

Amazon GameLift

These are the Read actions for Amazon GameLift.

- `gamelift:DescribeAlias`
- `gamelift:DescribeBuild`
- `gamelift:DescribeEC2InstanceLimits`
- `gamelift:DescribeFleetAttributes`
- `gamelift:DescribeFleetCapacity`
- `gamelift:DescribeFleetEvents`
- `gamelift:DescribeFleetPortSettings`
- `gamelift:DescribeFleetUtilization`
- `gamelift:DescribeGameSessionDetails`
- `gamelift:DescribeGameSessions`
- `gamelift:DescribeInstances`
- `gamelift:DescribePlayerSessions`
- `gamelift:DescribeRuntimeConfiguration`
- `gamelift:DescribeScalingPolicies`
- `gamelift:GetGameSessionLogUrl`
- `gamelift:GetInstanceAccess`
- `gamelift:RequestUploadCredentials`
- `gamelift:ResolveAlias`
- `gamelift:SearchGameSessions`

Amazon Glacier

These are the Read actions for Amazon Glacier.

- `glacier:DescribeJob`
- `glacier:DescribeVault`
- `glacier:GetDataRetrievalPolicy`
- `glacier:GetJobOutput`
- `glacier:GetVaultAccessPolicy`
- `glacier:GetVaultLock`
- `glacier:GetVaultNotifications`

AWS Health APIs and Notifications

These are the Read actions for AWS Health APIs and Notifications.

- `health:DescribeAffectedEntities`
- `health:DescribeEntityAggregates`
- `health:DescribeEventAggregates`
- `health:DescribeEventDetails`
- `health:DescribeEventTypes`
- `health:DescribeEvents`

Identity And Access Management

These are the Read actions for Identity And Access Management.

- `iam:GenerateCredentialReport`
- `iam:GenerateServiceLastAccessedDetails`
- `iam:GetAccessKeyLastUsed`
- `iam:GetAccountAuthorizationDetails`
- `iam:GetAccountPasswordPolicy`
- `iam:GetContextKeysForCustomPolicy`
- `iam:GetContextKeysForPrincipalPolicy`
- `iam:GetCredentialReport`
- `iam:GetGroup`
- `iam:GetGroupPolicy`
- `iamGetInstanceProfile`
- `iam:GetOpenIDConnectProvider`
- `iam:GetPolicy`
- `iam:GetPolicyVersion`
- `iam:GetRole`
- `iam:GetRolePolicy`
- `iam:GetSAMLProvider`
- `iam:GetSSHPublicKey`
- `iam:GetServerCertificate`
- `iam:GetServiceLastAccessedDetails`
- `iam:GetServiceLastAccessedDetailsWithEntities`
- `iam GetUser`
- `iam GetUserPolicy`
- `iam:SimulateCustomPolicy`
- `iam:SimulatePrincipalPolicy`

AWS Import Export Disk Service

These are the Read actions for AWS Import Export Disk Service.

- `importexport:GetShippingLabel`
- `importexport:GetStatus`

Amazon Inspector

These are the Read actions for Amazon Inspector.

- `inspector:DescribeAssessmentRuns`
- `inspector:DescribeAssessmentTargets`
- `inspector:DescribeAssessmentTemplates`
- `inspector:DescribeCrossAccountAccessRole`
- `inspector:DescribeFindings`
- `inspector:DescribeResourceGroups`
- `inspector:DescribeRulesPackages`
- `inspector:GetTelemetryMetadata`
- `inspector:PreviewAgents`

AWS IoT

These are the Read actions for AWS IoT.

- `iot:DescribeCACertificate`
- `iot:DescribeCertificate`
- `iot:DescribeEndpoint`
- `iot:DescribeThing`
- `iot:DescribeThingType`
- `iot:GetLoggingOptions`
- `iot:GetPolicy`
- `iot:GetPolicyVersion`
- `iot:GetRegistrationCode`
- `iot:GetThingShadow`
- `iot:GetTopicRule`

AWS Key Management Service

These are the Read actions for AWS Key Management Service.

- `kms:DescribeKey`
- `kms:GenerateRandom`
- `kms:GetKeyPolicy`
- `kms:GetKeyRotationStatus`
- `kms:GetParametersForImport`
- `kms>ListGrants`
- `kms>ListResourceTags`
- `kms:ReEncryptFrom`
- `kms:ReEncryptTo`

Amazon Kinesis

These are the Read actions for Amazon Kinesis.

- `kinesis:DescribeLimits`
- `kinesis:DescribeStream`
- `kinesis:GetRecords`
- `kinesis:GetShardIterator`

- `kinesis>ListTagsForStream`

Amazon Kinesis Analytics

These are the Read actions for Amazon Kinesis Analytics.

- `kinesisanalytics>DescribeApplication`
- `kinesisanalytics>DiscoverInputSchema`

AWS Lambda

These are the Read actions for AWS Lambda.

- `lambda>GetAccountSettings`
- `lambda>GetAlias`
- `lambda>GetEventSourceMapping`
- `lambda>GetFunction`
- `lambda>GetFunctionConfiguration`
- `lambda>GetPolicy`
- `lambda>ListTags`

Amazon Lex

These are the Read actions for Amazon Lex.

- `lex>GetBot`
- `lex>GetBotAlias`
- `lex>GetBotChannelAssociation`
- `lex>GetBuiltInIntent`
- `lex>GetBuiltInIntents`
- `lex>GetBuiltInSlotTypes`
- `lex>GetIntent`
- `lex>GetSlotType`

Amazon Lightsail

These are the Read actions for Amazon Lightsail.

- `lightsail>DownloadDefaultKeyPair`
- `lightsail>GetActiveNames`
- `lightsail>GetDomains`
- `lightsail>GetInstance`
- `lightsail>GetInstanceAccessDetails`
- `lightsail>GetInstanceMetricData`
- `lightsail>GetInstancePortStates`
- `lightsail>GetInstanceSnapshot`
- `lightsail>GetInstanceState`
- `lightsail>GetKeyPairs`
- `lightsail>GetOperation`

- `lightsail:GetOperations`
- `lightsail:GetOperationsForResource`
- `lightsail:GetStaticIp`

Amazon Machine Learning

These are the Read actions for Amazon Machine Learning.

- `machinelearning:GetBatchPrediction`
- `machinelearning:GetDataSource`
- `machinelearning:GetEvaluation`
- `machinelearning:GetMLModel`

Manage Amazon API Gateway

These are the Read actions for Manage Amazon API Gateway.

- `apigateway:GET`
- `apigateway:HEAD`
- `apigateway:OPTIONS`

Amazon Mechanical Turk

These are the Read actions for Amazon Mechanical Turk.

- `mechanicalturk:GetAccountBalance`
- `mechanicalturk:GetAssignment`
- `mechanicalturk:GetAssignmentsForHIT`
- `mechanicalturk:GetBlockedWorkers`
- `mechanicalturk:GetBonusPayments`
- `mechanicalturk:GetFileUploadURL`
- `mechanicalturk:GetHIT`
- `mechanicalturk:GetHITSForQualificationType`
- `mechanicalturk:GetQualificationRequests`
- `mechanicalturk:GetQualificationScore`
- `mechanicalturk:GetQualificationType`
- `mechanicalturk:GetQualificationsForQualificationType`
- `mechanicalturk:GetRequesterStatistic`
- `mechanicalturk:GetRequesterWorkerStatistic`
- `mechanicalturk:GetReviewResultsForHIT`
- `mechanicalturk:GetReviewableHITS`
- `mechanicalturk:SearchHITS`
- `mechanicalturk:SearchQualificationTypes`

Amazon Message Delivery Service

These are the Read actions for Amazon Message Delivery Service.

- `ec2messages:GetEndpoint`

- `ec2messages:GetMessages`

Amazon Mobile Analytics

These are the Read actions for Amazon Mobile Analytics.

- `mobileanalytics:GetFinancialReports`
- `mobileanalytics:GetReports`

AWS Mobile Hub

These are the Read actions for AWS Mobile Hub.

- `mobilehub:GetProject`
- `mobilehub:ValidateProject`
- `mobilehub:VerifyServiceRole`

AWS OpsWorks

These are the Read actions for AWS OpsWorks.

- `opsworks:GetHostnameSuggestion`

AWS Organizations

These are the Read actions for AWS Organizations.

- `organizations:DescribeAccount`
- `organizations:DescribeCreateAccountStatus`
- `organizations:DescribeHandshake`
- `organizations:DescribeOrganization`
- `organizations:DescribeOrganizationalUnit`
- `organizations:DescribePolicy`

Amazon Pinpoint

These are the Read actions for Amazon Pinpoint.

- `mobiletargeting:GetApnsChannel`
- `mobiletargeting:GetCampaign`
- `mobiletargeting:GetCampaignActivities`
- `mobiletargeting:GetCampaignVersion`
- `mobiletargeting:GetCampaignVersions`
- `mobiletargeting:GetEndpoint`
- `mobiletargeting:GetGcmChannel`
- `mobiletargeting:ImportJob`
- `mobiletargeting:GetReports`
- `mobiletargeting:GetSegment`
- `mobiletargeting:SegmentImportJobs`

- `mobiletargeting:GetSegmentVersion`
- `mobiletargeting:GetSegmentVersions`

Amazon Polly

These are the Read actions for Amazon Polly.

- `polly:GetLexicon`
- `polly:SynthesizeSpeech`

Amazon RDS

These are the Read actions for Amazon RDS.

- `rds:DescribeDBClusterSnapshots`
- `rds:DownloadCompleteDBLogFile`
- `rds:DownloadDBLogFilePortion`
- `rds>ListTagsForResource`

Amazon Redshift

These are the Read actions for Amazon Redshift.

- `redshift:DescribeClusterParameterGroups`
- `redshift:DescribeClusterParameters`
- `redshift:DescribeClusterSecurityGroups`
- `redshift:DescribeClusterSnapshots`
- `redshift:DescribeClusterSubnetGroups`
- `redshift:DescribeClusterVersions`
- `redshift:DescribeDefaultClusterParameters`
- `redshift:DescribeEventCategories`
- `redshift:DescribeEventSubscriptions`
- `redshift:DescribeHsmClientCertificates`
- `redshift:DescribeHsmConfigurations`
- `redshift:DescribeLoggingStatus`
- `redshift:DescribeOrderableClusterOptions`
- `redshift:DescribeReservedNodeOfferings`
- `redshift:DescribeReservedNodes`
- `redshift:DescribeResize`
- `redshift:DescribeSnapshotCopyGrants`
- `redshift:DescribeTableRestoreStatus`
- `redshift:DescribeTags`
- `redshift:GetClusterCredentials`

Amazon Rekognition

These are the Read actions for Amazon Rekognition.

- `rekognition:CompareFaces`

- rekognition:DetectFaces
- rekognition:DetectLabels
- rekognition:DetectModerationLabels
- rekognition>ListCollections
- rekognition>ListFaces
- rekognition:SearchFaces
- rekognition:SearchFacesByImage

Amazon Resource Group Tagging API

These are the Read actions for Amazon Resource Group Tagging API.

- tag:GetResources
- tag:GetTagKeys
- tag:GetTagValues

Amazon Route 53

These are the Read actions for Amazon Route 53.

- route53:GetHealthCheck
- route53:GetTrafficPolicy
- route53:GetTrafficPolicyInstance
- route53:GetTrafficPolicyInstanceCount
- route53>ListHealthChecks
- route53>ListTrafficPolicies
- route53>ListTrafficPolicyInstances
- route53>ListTrafficPolicyInstancesByHostedZone
- route53>ListTrafficPolicyInstancesByPolicy
- route53>ListTrafficPolicyVersions
- route53:TestDNSAnswer

Amazon Route53 Domains

These are the Read actions for Amazon Route53 Domains.

- route53domains:CheckDomainAvailability
- route53domains:GetContactReachabilityStatus
- route53domains:GetDomainDetail
- route53domains:GetDomainSuggestions
- route53domains:GetOperationDetail
- route53domains:ViewBilling

Amazon S3

These are the Read actions for Amazon S3.

- s3:GetAccelerateConfiguration
- s3:GetAnalyticsConfiguration

- `s3:GetBucketAcl`
- `s3:GetBucketCORS`
- `s3:GetBucketLocation`
- `s3:GetBucketLogging`
- `s3:GetBucketNotification`
- `s3:GetBucketPolicy`
- `s3:GetBucketRequestPayment`
- `s3:GetBucketTagging`
- `s3:GetBucketVersioning`
- `s3:GetBucketWebsite`
- `s3:GetInventoryConfiguration`
- `s3:GetIpConfiguration`
- `s3:GetLifecycleConfiguration`
- `s3:GetMetricsConfiguration`
- `s3:GetObject`
- `s3:GetObjectAcl`
- `s3:GetObjectTagging`
- `s3:GetObjectTorrent`
- `s3:GetObjectVersion`
- `s3:GetObjectVersionAcl`
- `s3:GetObjectVersionForReplication`
- `s3:GetObjectVersionTagging`
- `s3:GetObjectVersionTorrent`
- `s3:GetReplicationConfiguration`
- `s3>ListBucketByTags`
- `s3>ListBucketMultipartUploads`
- `s3>ListBucketVersions`
- `s3>ListMultipartUploadParts`

Amazon SES

These are the Read actions for Amazon SES.

- `ses:DescribeActiveReceiptRuleSet`
- `ses:DescribeReceiptRule`
- `ses:DescribeReceiptRuleSet`
- `ses:GetIdentityDkimAttributes`
- `ses:GetIdentityMailFromDomainAttributes`
- `ses:GetIdentityNotificationAttributes`
- `ses:GetIdentityPolicies`
- `ses:GetIdentityVerificationAttributes`
- `ses:GetSendQuota`
- `ses:GetSendStatistics`
- `ses:VerifyDomainDkim`
- `ses:VerifyDomainIdentity`
- `ses:VerifyEmailAddress`

- `ses:VerifyEmailIdentity`

Amazon SNS

These are the Read actions for Amazon SNS.

- `sns:CheckIfPhoneNumberIsOptedOut`
- `sns:GetEndpointAttributes`
- `sns:GetPlatformApplicationAttributes`
- `sns:GetSMSAttributes`
- `sns:GetSubscriptionAttributes`
- `sns:GetTopicAttributes`
- `sns>ListPhoneNumbersOptedOut`

Amazon SQS

These are the Read actions for Amazon SQS.

- `sqs:GetQueueAttributes`
- `sqs:GetQueueUrl`
- `sqs>ListDeadLetterSourceQueues`
- `sqs:ReceiveMessage`

AWS Security Token Service

These are the Read actions for AWS Security Token Service.

- `sts:GetCallerIdentity`
- `sts:GetFederationToken`
- `sts:GetSessionToken`

AWS Service Catalog

These are the Read actions for AWS Service Catalog.

- `servicecatalog:DescribeConstraint`
- `servicecatalog:DescribePortfolio`
- `servicecatalog:DescribeProduct`
- `servicecatalog:DescribeProductAsAdmin`
- `servicecatalog:DescribeProductView`
- `servicecatalog:DescribeProvisioningArtifact`
- `servicecatalog:DescribeProvisioningParameters`
- `servicecatalog:DescribeRecord`

AWS Shield

These are the Read actions for AWS Shield.

- `shield:DescribeAttack`

- `shield:DescribeProtection`
- `shield:DescribeSubscription`

Amazon Simple Systems Manager

These are the Read actions for Amazon Simple Systems Manager.

- `ssm:DescribeActivations`
- `ssm:DescribeAssociation`
- `ssm:DescribeAutomationActions`
- `ssm:DescribeAutomationExecutions`
- `ssm:DescribeAvailablePatches`
- `ssm:DescribeDocument`
- `ssm:DescribeDocumentPermission`
- `ssm:DescribeEffectiveInstanceAssociations`
- `ssm:DescribeEffectivePatchesForPatchBaseline`
- `ssm:DescribeInstanceAssociationsStatus`
- `ssm:DescribeInstanceInformation`
- `ssm:DescribeInstancePatchStates`
- `ssm:DescribeInstancePatchStatesForPatchGroup`
- `ssm:DescribeInstancePatches`
- `ssm:DescribeMaintenanceWindowExecutionTaskInvocations`
- `ssm:DescribeMaintenanceWindowExecutionTasks`
- `ssm:DescribeMaintenanceWindowExecutions`
- `ssm:DescribeMaintenanceWindowTargets`
- `ssm:DescribeMaintenanceWindowTasks`
- `ssm:DescribeMaintenanceWindows`
- `ssm:DescribeParameters`
- `ssm:DescribePatchBaselines`
- `ssm:DescribePatchGroupState`
- `ssm:DescribePatchGroups`
- `ssm:GetCommandInvocation`
- `ssm:GetDefaultPatchBaseline`
- `ssm:GetDeployablePatchSnapshotForInstance`
- `ssm:GetDocument`
- `ssm:GetInventory`
- `ssm:GetInventorySchema`
- `ssm:GetMaintenanceWindow`
- `ssm:GetMaintenanceWindowExecution`
- `ssm:GetMaintenanceWindowExecutionTask`
- `ssm:GetParameterHistory`
- `ssm:GetParameters`
- `ssm:GetPatchBaseline`
- `ssm:GetPatchBaselineForPatchGroup`
- `ssm>ListCommandInvocations`
- `ssm>ListCommands`

- `ssm>ListTagsForResource`

Amazon Simple Workflow Service

These are the Read actions for Amazon Simple Workflow Service.

- `swf:CountClosedWorkflowExecutions`
- `swf:CountOpenWorkflowExecutions`
- `swf:CountPendingActivityTasks`
- `swf:CountPendingDecisionTasks`
- `swf:DescribeActivityType`
- `swf:DescribeDomain`
- `swf:DescribeWorkflowExecution`
- `swf:DescribeWorkflowType`
- `swf:GetWorkflowExecutionHistory`

Amazon SimpleDB

These are the Read actions for Amazon SimpleDB.

- `sdb:GetAttributes`
- `sdb:Select`

AWS Step Functions

These are the Read actions for AWS Step Functions.

- `states:DescribeActivity`
- `states:DescribeExecution`
- `states:DescribeStateMachine`
- `states:GetExecutionHistory`
- `states>ListExecutions`

Amazon Storage Gateway

These are the Read actions for Amazon Storage Gateway.

- `storagegateway:DescribeBandwidthRateLimit`
- `storagegateway:DescribeCache`
- `storagegateway:DescribeCachediSCSIVolumes`
- `storagegateway:DescribeChapCredentials`
- `storagegateway:DescribeGatewayInformation`
- `storagegateway:DescribeMaintenanceStartTime`
- `storagegateway:DescribeSnapshotSchedule`
- `storagegateway:DescribeStorediSCSIVolumes`
- `storagegateway:DescribeTapeArchives`
- `storagegateway:DescribeTapeRecoveryPoints`
- `storagegateway:DescribeTapes`
- `storagegateway:DescribeUploadBuffer`

- `storagegateway:DescribeVTLDevices`
- `storagegateway:DescribeWorkingStorage`
- `storagegateway>ListTagsForResource`
- `storagegateway>ListTapes`
- `storagegateway>ListVolumeInitiators`

AWS WAF

These are the Read actions for AWS WAF.

- `waf:GetByteMatchSet`
- `waf:GetChangeToken`
- `waf:GetChangeTokenStatus`
- `waf:GetIPSet`
- `waf:GetRateBasedRule`
- `waf:GetRateBasedRuleManagedKeys`
- `waf:GetRule`
- `waf:GetSampledRequests`
- `waf:GetSizeConstraintSet`
- `waf:GetSqlInjectionMatchSet`
- `waf:GetWebACL`
- `waf:GetXssMatchSet`

AWS WAF Regional

These are the Read actions for AWS WAF Regional.

- `waf-regional:GetByteMatchSet`
- `waf-regional:GetChangeToken`
- `waf-regional:GetChangeTokenStatus`
- `waf-regional:GetIPSet`
- `waf-regional:GetRateBasedRule`
- `waf-regional:GetRateBasedRuleManagedKeys`
- `waf-regional:GetRule`
- `waf-regional:GetSampledRequests`
- `waf-regional:GetSizeConstraintSet`
- `waf-regional:GetSqlInjectionMatchSet`
- `waf-regional:GetWebACL`
- `waf-regional:GetWebACLForResource`
- `waf-regional:GetXssMatchSet`

Amazon WorkDocs

These are the Read actions for Amazon WorkDocs.

- `workdocs:CheckAlias`
- `workdocs:GetDocument`
- `workdocs:GetDocumentPath`

- `workdocs:GetDocumentVersion`
- `workdocs:GetFolder`
- `workdocs:GetFolderPath`

Amazon WorkMail

These are the Read actions for Amazon WorkMail.

- `workmail:GetMailDomainDetails`
- `workmail:GetMailGroupDetails`
- `workmail:GetMailUserDetails`
- `workmail:GetMobileDeviceDetails`
- `workmail:GetMobileDevicesForUser`
- `workmail:GetMobilePolicyDetails`
- `workmail>ListMembersInMailGroup`
- `workmail:SearchMembers`

Amazon WorkSpaces

These are the Read actions for Amazon WorkSpaces.

- `workspaces:DescribeWorkspacesConnectionStatus`

AWS XRay

These are the Read actions for AWS XRay.

- `xray:BatchGetTraces`
- `xray:GetServiceGraph`
- `xray:GetTraceGraph`
- `xray:GetTraceSummaries`

Service Actions Included in the Write Access Level

You can use the actions below to create, modify, or delete AWS resources. If an AWS service does not appear on this page then that service does not have any actions in the Write category.

Topics

- [Amazon API Gateway \(p. 665\)](#)
- [Amazon AWS Cloud Contact Center \(p. 666\)](#)
- [Amazon AppStream \(p. 666\)](#)
- [Application Auto Scaling \(p. 666\)](#)
- [Application Discovery \(p. 666\)](#)
- [Amazon Athena \(p. 667\)](#)
- [Auto Scaling \(p. 667\)](#)
- [AWS Batch \(p. 668\)](#)
- [AWS Billing \(p. 668\)](#)
- [AWS Budget Service \(p. 668\)](#)
- [AWS Certificate Manager \(p. 668\)](#)

- [Amazon Cloud Directory \(p. 668\)](#)
- [AWS CloudFormation \(p. 669\)](#)
- [Amazon CloudFront \(p. 670\)](#)
- [AWS CloudHSM \(p. 670\)](#)
- [Amazon CloudSearch \(p. 670\)](#)
- [AWS CloudTrail \(p. 671\)](#)
- [Amazon CloudWatch \(p. 671\)](#)
- [Amazon CloudWatch Events \(p. 671\)](#)
- [Amazon CloudWatch Logs \(p. 671\)](#)
- [AWS CodeBuild \(p. 672\)](#)
- [AWS CodeCommit \(p. 672\)](#)
- [AWS CodeDeploy \(p. 672\)](#)
- [AWS CodePipeline \(p. 673\)](#)
- [AWS CodeStar \(p. 673\)](#)
- [Amazon Cognito Identity \(p. 674\)](#)
- [Amazon Cognito Sync \(p. 674\)](#)
- [Amazon Cognito User Pools \(p. 674\)](#)
- [AWS Config \(p. 675\)](#)
- [AWS Cost and Usage Report \(p. 676\)](#)
- [Data Pipeline \(p. 676\)](#)
- [AWS Database Migration Service \(p. 676\)](#)
- [AWS Device Farm \(p. 677\)](#)
- [AWS Direct Connect \(p. 677\)](#)
- [AWS Directory Service \(p. 678\)](#)
- [Amazon DynamoDB \(p. 678\)](#)
- [Amazon DynamoDB Accelerator \(DAX\) \(p. 679\)](#)
- [Amazon EC2 \(p. 679\)](#)
- [Amazon EC2 Container Registry \(p. 683\)](#)
- [Amazon EC2 Container Service \(p. 683\)](#)
- [Amazon ElastiCache \(p. 683\)](#)
- [AWS Elastic Beanstalk \(p. 684\)](#)
- [Amazon Elastic File System \(p. 685\)](#)
- [Elastic Load Balancing V2 \(p. 685\)](#)
- [Amazon Elastic MapReduce \(p. 685\)](#)
- [Amazon Elastic Transcoder \(p. 686\)](#)
- [Amazon Elasticsearch Service \(p. 686\)](#)
- [Amazon GameLift \(p. 686\)](#)
- [Amazon Glacier \(p. 687\)](#)
- [Identity And Access Management \(p. 687\)](#)
- [AWS Import Export Disk Service \(p. 688\)](#)
- [Amazon Inspector \(p. 689\)](#)
- [AWS IoT \(p. 689\)](#)
- [AWS Key Management Service \(p. 690\)](#)
- [Amazon Kinesis \(p. 690\)](#)
- [Amazon Kinesis Analytics \(p. 691\)](#)
- [Amazon Kinesis Firehose \(p. 691\)](#)

- [AWS Lambda \(p. 691\)](#)
- [Amazon Lex \(p. 692\)](#)
- [Amazon Lightsail \(p. 692\)](#)
- [Amazon Machine Learning \(p. 693\)](#)
- [Manage Amazon API Gateway \(p. 693\)](#)
- [AWS Marketplace \(p. 693\)](#)
- [AWS Marketplace Management Portal \(p. 694\)](#)
- [Amazon Mechanical Turk \(p. 694\)](#)
- [Amazon Message Delivery Service \(p. 694\)](#)
- [Amazon Mobile Analytics \(p. 695\)](#)
- [AWS Mobile Hub \(p. 695\)](#)
- [AWS OpsWorks \(p. 695\)](#)
- [AWS OpsWorks Configuration Management \(p. 696\)](#)
- [AWS Organizations \(p. 696\)](#)
- [Amazon Pinpoint \(p. 697\)](#)
- [Amazon Polly \(p. 697\)](#)
- [Amazon RDS \(p. 697\)](#)
- [Amazon Redshift \(p. 699\)](#)
- [Amazon Rekognition \(p. 700\)](#)
- [Amazon Resource Group Tagging API \(p. 700\)](#)
- [Amazon Route 53 \(p. 700\)](#)
- [Amazon Route53 Domains \(p. 700\)](#)
- [Amazon S3 \(p. 701\)](#)
- [Amazon SES \(p. 702\)](#)
- [Amazon SNS \(p. 702\)](#)
- [Amazon SQS \(p. 703\)](#)
- [AWS Security Token Service \(p. 703\)](#)
- [AWS Service Catalog \(p. 703\)](#)
- [AWS Shield \(p. 704\)](#)
- [Amazon Simple Systems Manager \(p. 704\)](#)
- [Amazon Simple Workflow Service \(p. 705\)](#)
- [Amazon SimpleDB \(p. 705\)](#)
- [AWS Step Functions \(p. 706\)](#)
- [Amazon Storage Gateway \(p. 706\)](#)
- [AWS Trusted Advisor \(p. 707\)](#)
- [AWS WAF \(p. 707\)](#)
- [AWS WAF Regional \(p. 708\)](#)
- [Amazon WorkDocs \(p. 708\)](#)
- [Amazon WorkMail \(p. 709\)](#)
- [Amazon WorkSpaces \(p. 709\)](#)
- [Amazon WorkSpaces Application Manager \(p. 710\)](#)
- [AWS XRay \(p. 710\)](#)

Amazon API Gateway

These are the Write actions for Amazon API Gateway.

- `execute-api:InvalidateCache`
- `execute-api:Invoke`

Amazon AWS Cloud Contact Center

These are the Write actions for Amazon AWS Cloud Contact Center.

- `connect>CreateInstance`
- `connect:DestroyInstance`
- `connect:ModifyInstance`

Amazon AppStream

These are the Write actions for Amazon AppStream.

- `appstream:AssociateFleet`
- `appstream:CreateFleet`
- `appstream:CreateStack`
- `appstream>CreateStreamingURL`
- `appstream:DeleteFleet`
- `appstream:DeleteStack`
- `appstream:DisassociateFleet`
- `appstream:ExpireSession`
- `appstream:StartFleet`
- `appstream:StopFleet`
- `appstream:UpdateFleet`
- `appstream:UpdateStack`

Application Auto Scaling

These are the Write actions for Application Auto Scaling.

- `application-autoscaling>DeleteScalingPolicy`
- `application-autoscaling:DeregisterScalableTarget`
- `application-autoscaling:PutScalingPolicy`
- `application-autoscaling:RegisterScalableTarget`

Application Discovery

These are the Write actions for Application Discovery.

- `discovery:AssociateConfigurationItemsToApplication`
- `discovery>CreateApplication`
- `discovery>CreateTags`
- `discovery>DeleteApplications`
- `discovery>DeleteTags`
- `discovery:DisassociateConfigurationItemsFromApplication`
- `discovery:ExportConfigurations`

- `discovery:StartDataCollectionByAgentIds`
- `discovery:StartExportTask`
- `discovery:StopDataCollectionByAgentIds`
- `discovery:UpdateApplication`

Amazon Athena

These are the Write actions for Amazon Athena.

- `athena:CreateNamedQuery`
- `athena:DeleteNamedQuery`
- `athena:StartQueryExecution`
- `athena:StopQueryExecution`

Auto Scaling

These are the Write actions for Auto Scaling.

- `autoscaling:AttachInstances`
- `autoscaling:AttachLoadBalancerTargetGroups`
- `autoscaling:AttachLoadBalancers`
- `autoscaling:CompleteLifecycleAction`
- `autoscaling>CreateAutoScalingGroup`
- `autoscaling>CreateLaunchConfiguration`
- `autoscaling>CreateOrUpdateTags`
- `autoscaling>DeleteAutoScalingGroup`
- `autoscaling>DeleteLaunchConfiguration`
- `autoscaling>DeleteLifecycleHook`
- `autoscaling>DeleteNotificationConfiguration`
- `autoscaling>DeleteScheduledAction`
- `autoscaling>DeleteTags`
- `autoscaling:DetachInstances`
- `autoscaling:DetachLoadBalancers`
- `autoscaling:DisableMetricsCollection`
- `autoscaling:EnableMetricsCollection`
- `autoscaling:EnterStandby`
- `autoscaling:ExitStandby`
- `autoscaling:PutLifecycleHook`
- `autoscaling:PutNotificationConfiguration`
- `autoscaling:PutScheduledUpdateGroupAction`
- `autoscaling:RecordLifecycleActionHeartbeat`
- `autoscaling:ResumeProcesses`
- `autoscaling:SetDesiredCapacity`
- `autoscaling:SetInstanceHealth`
- `autoscaling:SetInstanceProtection`
- `autoscaling:SuspendProcesses`

- `autoscaling:TerminateInstanceInAutoScalingGroup`
- `autoscaling:UpdateAutoScalingGroup`

AWS Batch

These are the Write actions for AWS Batch.

- `batch:CancelJob`
- `batch>CreateComputeEnvironment`
- `batch>CreateJobQueue`
- `batch>DeleteComputeEnvironment`
- `batch>DeleteJobQueue`
- `batch:DeregisterJobDefinition`
- `batch:RegisterJobDefinition`
- `batch:SubmitJob`
- `batch:TerminateJob`
- `batch:UpdateComputeEnvironment`
- `batch:UpdateJobQueue`

AWS Billing

These are the Write actions for AWS Billing.

- `aws-portal:ModifyAccount`
- `aws-portal:ModifyBilling`
- `aws-portal:ModifyPaymentMethods`

AWS Budget Service

These are the Write actions for AWS Budget Service.

- `budgets:ModifyBudget`

AWS Certificate Manager

These are the Write actions for AWS Certificate Manager.

- `acm:AddTagsToCertificate`
- `acm:ImportCertificate`
- `acm:RemoveTagsFromCertificate`
- `acm:RequestCertificate`

Amazon Cloud Directory

These are the Write actions for Amazon Cloud Directory.

- `clouddirectory:AddFacetToObject`
- `clouddirectory:ApplySchema`
- `clouddirectory:AttachObject`

- `clouddirectory:AttachPolicy`
- `clouddirectory:AttachToIndex`
- `clouddirectory:AttachTypedLink`
- `clouddirectory:BatchWrite`
- `clouddirectory>CreateDirectory`
- `clouddirectory>CreateFacet`
- `clouddirectory>CreateIndex`
- `clouddirectory>CreateObject`
- `clouddirectory>CreateSchema`
- `clouddirectory>CreateTypedLinkFacet`
- `clouddirectory>DeleteDirectory`
- `clouddirectory>DeleteFacet`
- `clouddirectory>DeleteObject`
- `clouddirectory>DeleteSchema`
- `clouddirectory>DeleteTypedLinkFacet`
- `clouddirectory:DetachFromIndex`
- `clouddirectory:DetachObject`
- `clouddirectory:DetachPolicy`
- `clouddirectory:DetachTypedLink`
- `clouddirectory:DisableDirectory`
- `clouddirectory:EnableDirectory`
- `clouddirectory:PublishSchema`
- `clouddirectory:PutSchemaFromJson`
- `clouddirectory:RemoveFacetFromObject`
- `clouddirectory:TagResource`
- `clouddirectory:UntagResource`
- `clouddirectory:UpdateFacet`
- `clouddirectory:UpdateObjectAttributes`
- `clouddirectory:UpdateSchema`
- `clouddirectory:UpdateTypedLinkFacet`

AWS CloudFormation

These are the Write actions for AWS CloudFormation.

- `cloudformation:CancelUpdateStack`
- `cloudformation:ContinueUpdateRollback`
- `cloudformation>CreateChangeSet`
- `cloudformation>CreateStack`
- `cloudformation>CreateUploadBucket`
- `cloudformation>DeleteChangeSet`
- `cloudformation>DeleteStack`
- `cloudformation:ExecuteChangeSet`
- `cloudformation:SignalResource`
- `cloudformation:UpdateStack`
- `cloudformation:ValidateTemplate`

Amazon CloudFront

These are the Write actions for Amazon CloudFront.

- `cloudfront:CreateCloudFrontOriginAccessIdentity`
- `cloudfront:CreateDistribution`
- `cloudfront:CreateDistributionWithTags`
- `cloudfront:CreateInvalidation`
- `cloudfront:CreateStreamingDistribution`
- `cloudfront:CreateStreamingDistributionWithTags`
- `cloudfront:DeleteCloudFrontOriginAccessIdentity`
- `cloudfront:DeleteDistribution`
- `cloudfront:DeleteStreamingDistribution`
- `cloudfront:TagResource`
- `cloudfront:UntagResource`
- `cloudfront:UpdateCloudFrontOriginAccessIdentity`
- `cloudfront:UpdateDistribution`
- `cloudfront:UpdateStreamingDistribution`

AWS CloudHSM

These are the Write actions for AWS CloudHSM.

- `clouhdsm:AddTagsToResource`
- `clouhdsm>CreateHapg`
- `clouhdsm>CreateHsm`
- `clouhdsm>CreateLunaClient`
- `clouhdsm>DeleteHapg`
- `clouhdsm>DeleteHsm`
- `clouhdsm>DeleteLunaClient`
- `clouhdsm:ModifyHapg`
- `clouhdsm:ModifyHsm`
- `clouhdsm:ModifyLunaClient`
- `clouhdsm:RemoveTagsFromResource`

Amazon CloudSearch

These are the Write actions for Amazon CloudSearch.

- `cloudsearch:AddTags`
- `cloudsearch:BuildSuggesters`
- `cloudsearch>CreateDomain`
- `cloudsearch:DefineAnalysisScheme`
- `cloudsearch:DefineExpression`
- `cloudsearch:DefineIndexField`
- `cloudsearch:DefineSuggester`
- `cloudsearch>DeleteAnalysisScheme`
- `cloudsearch>DeleteDomain`

- `cloudsearch>DeleteExpression`
- `cloudsearch>DeleteIndexField`
- `cloudsearch>DeleteSuggester`
- `cloudsearch>IndexDocuments`
- `cloudsearch>RemoveTags`
- `cloudsearch>UpdateAvailabilityOptions`
- `cloudsearch>UpdateScalingParameters`
- `cloudsearch>document`

AWS CloudTrail

These are the Write actions for AWS CloudTrail.

- `cloudtrail>AddTags`
- `cloudtrail>CreateTrail`
- `cloudtrail>DeleteTrail`
- `cloudtrail>PutEventSelectors`
- `cloudtrail>RemoveTags`
- `cloudtrail>StartLogging`
- `cloudtrail>StopLogging`
- `cloudtrail>UpdateTrail`

Amazon CloudWatch

These are the Write actions for Amazon CloudWatch.

- `cloudwatch>DeleteAlarms`
- `cloudwatch>DisableAlarmActions`
- `cloudwatch>EnableAlarmActions`
- `cloudwatch>PutMetricAlarm`
- `cloudwatch>PutMetricData`
- `cloudwatch>SetAlarmState`

Amazon CloudWatch Events

These are the Write actions for Amazon CloudWatch Events.

- `events>DeleteRule`
- `events>DisableRule`
- `events>EnableRule`
- `events>PutEvents`
- `events>PutRule`
- `events>PutTargets`
- `events>RemoveTargets`

Amazon CloudWatch Logs

These are the Write actions for Amazon CloudWatch Logs.

- logs:CancelExportTask
- logs>CreateExportTask
- logs>CreateLogGroup
- logs>CreateLogStream
- logs>DeleteDestination
- logs>DeleteLogGroup
- logs>DeleteLogStream
- logs>DeleteMetricFilter
- logs>DeleteRetentionPolicy
- logs>DeleteSubscriptionFilter
- logs>PutDestination
- logs>PutDestinationPolicy
- logs>PutLogEvents
- logs>PutMetricFilter
- logs>PutRetentionPolicy
- logs>PutSubscriptionFilter

AWS CodeBuild

These are the Write actions for AWS CodeBuild.

- codebuild>CreateProject
- codebuild>DeleteProject
- codebuild>PersistOAuthToken
- codebuild>StartBuild
- codebuild>StopBuild
- codebuild>UpdateProject

AWS CodeCommit

These are the Write actions for AWS CodeCommit.

- codecommit>CreateBranch
- codecommit>CreateRepository
- codecommit>DeleteBranch
- codecommit>DeleteRepository
- codecommit>GitPush
- codecommit>PutRepositoryTriggers
- codecommit>TestRepositoryTriggers
- codecommit>UpdateDefaultBranch
- codecommit>UpdateRepositoryDescription
- codecommit>UpdateRepositoryName

AWS CodeDeploy

These are the Write actions for AWS CodeDeploy.

- codedeploy>AddTagsToOnPremisesInstances

- `codedeploy:ContinueDeployment`
- `codedeploy>CreateApplication`
- `codedeploy>CreateDeployment`
- `codedeploy>CreateDeploymentConfig`
- `codedeploy>CreateDeploymentGroup`
- `codedeploy>DeleteApplication`
- `codedeploy>DeleteDeploymentConfig`
- `codedeploy>DeleteDeploymentGroup`
- `codedeploy>DeregisterOnPremisesInstance`
- `codedeploy:RegisterApplicationRevision`
- `codedeploy:RegisterOnPremisesInstance`
- `codedeploy:RemoveTagsFromOnPremisesInstances`
- `codedeploy:StopDeployment`
- `codedeploy:UpdateApplication`
- `codedeploy:UpdateDeploymentGroup`

AWS CodePipeline

These are the Write actions for AWS CodePipeline.

- `codepipeline:AcknowledgeJob`
- `codepipeline:AcknowledgeThirdPartyJob`
- `codepipeline>CreateCustomActionType`
- `codepipeline>CreatePipeline`
- `codepipeline>DeleteCustomActionType`
- `codepipeline>DeletePipeline`
- `codepipeline:DisableStageTransition`
- `codepipeline:EnableStageTransition`
- `codepipeline:PollForJobs`
- `codepipeline:PollForThirdPartyJobs`
- `codepipeline:PutActionRevision`
- `codepipeline:PutApprovalResult`
- `codepipeline:PutJobFailureResult`
- `codepipeline:PutJobSuccessResult`
- `codepipeline:PutThirdPartyJobFailureResult`
- `codepipeline:PutThirdPartyJobSuccessResult`
- `codepipeline:RetryStageExecution`
- `codepipeline:StartPipelineExecution`
- `codepipeline:UpdatePipeline`

AWS CodeStar

These are the Write actions for AWS CodeStar.

- `codestar>CreateUserProfile`
- `codestar>DeleteExtendedAccess`
- `codestar>DeleteUserProfile`

- `codestar:PutExtendedAccess`
- `codestar:UpdateProject`
- `codestar:UpdateUserProfile`

Amazon Cognito Identity

These are the Write actions for Amazon Cognito Identity.

- `cognito-identity:CreateIdentityPool`
- `cognito-identity:DeleteIdentities`
- `cognito-identity:DeleteIdentityPool`
- `cognito-identity:GetId`
- `cognito-identity:MergeDeveloperIdentities`
- `cognito-identity:SetIdentityPoolRoles`
- `cognito-identity:UnlinkDeveloperIdentity`
- `cognito-identity:UnlinkIdentity`
- `cognito-identity:UpdateIdentityPool`

Amazon Cognito Sync

These are the Write actions for Amazon Cognito Sync.

- `cognito-sync:BulkPublish`
- `cognito-sync:DeleteDataset`
- `cognito-sync:RegisterDevice`
- `cognito-sync:SetCognitoEvents`
- `cognito-sync:SetDatasetConfiguration`
- `cognito-sync:SetIdentityPoolConfiguration`
- `cognito-sync:SubscribeToDataset`
- `cognito-sync:UnsubscribeFromDataset`
- `cognito-sync:UpdateRecords`

Amazon Cognito User Pools

These are the Write actions for Amazon Cognito User Pools.

- `cognito-idp:AddCustomAttributes`
- `cognito-idp:AdminAddUserToGroup`
- `cognito-idp:AdminConfirmSignUp`
- `cognito-idp:AdminCreateUser`
- `cognito-idp:AdminDeleteUser`
- `cognito-idp:AdminDeleteUserAttributes`
- `cognito-idp:AdminDisableUser`
- `cognito-idp:AdminEnableUser`
- `cognito-idp:AdminForgetDevice`
- `cognito-idp:AdminInitiateAuth`
- `cognito-idp:AdminRemoveUserFromGroup`

- `cognito-idp:AdminResetUserPassword`
- `cognito-idp:AdminRespondToAuthChallenge`
- `cognito-idp:AdminSetUserSettings`
- `cognito-idp:AdminUpdateDeviceStatus`
- `cognito-idp:AdminUpdateUserAttributes`
- `cognito-idp:AdminUserGlobalSignOut`
- `cognito-idp:ChangePassword`
- `cognito-idp:ConfirmDevice`
- `cognito-idp:ConfirmForgotPassword`
- `cognito-idp:ConfirmSignUp`
- `cognito-idp>CreateGroup`
- `cognito-idp:CreateUserImportJob`
- `cognito-idp:CreateUserPool`
- `cognito-idp:CreateUserPoolClient`
- `cognito-idp>DeleteGroup`
- `cognito-idp>DeleteUser`
- `cognito-idp>DeleteUserAttributes`
- `cognito-idp>DeleteUserPool`
- `cognito-idp>DeleteUserPoolClient`
- `cognito-idp:ForgetDevice`
- `cognito-idp:ForgotPassword`
- `cognito-idp:GlobalSignOut`
- `cognito-idp:InitiateAuth`
- `cognito-idp:ResendConfirmationCode`
- `cognito-idp:RespondToAuthChallenge`
- `cognito-idp:SetUserSettings`
- `cognito-idp:SignUp`
- `cognito-idp:StartUserImportJob`
- `cognito-idp:StopUserImportJob`
- `cognito-idp:UpdateDeviceStatus`
- `cognito-idp:UpdateGroup`
- `cognito-idp:UpdateUserAttributes`
- `cognito-idp:UpdateUserPool`
- `cognito-idp:UpdateUserPoolClient`
- `cognito-idp:VerifyUserAttribute`

AWS Config

These are the Write actions for AWS Config.

- `config:DeleteConfigRule`
- `config:DeleteConfigurationRecorder`
- `config:DeleteDeliveryChannel`
- `config:DeleteEvaluationResults`
- `config:PutConfigRule`
- `config:PutConfigurationRecorder`

- config:PutDeliveryChannel
- config:PutEvaluations
- config:StartConfigRulesEvaluation
- config:StartConfigurationRecorder
- config:StopConfigurationRecorder

AWS Cost and Usage Report

These are the Write actions for AWS Cost and Usage Report.

- cur:DeleteReportDefinition
- cur:PutReportDefinition

Data Pipeline

These are the Write actions for Data Pipeline.

- datapipeline:ActivatePipeline
- datapipeline:AddTags
- datapipeline>CreatePipeline
- datapipeline:DeactivatePipeline
- datapipeline>DeletePipeline
- datapipeline:PollForTask
- datapipeline:PutAccountLimits
- datapipeline:PutPipelineDefinition
- datapipeline:RemoveTags
- datapipeline:ReportTaskProgress
- datapipeline:ReportTaskRunnerHeartbeat
- datapipeline:SetStatus
- datapipeline:SetTaskStatus

AWS Database Migration Service

These are the Write actions for AWS Database Migration Service.

- dms:AddTagsToResource
- dms:CreateEndpoint
- dms:CreateReplicationInstance
- dms:CreateReplicationSubnetGroup
- dms:CreateReplicationTask
- dms:DeleteEndpoint
- dms:DeleteEventSubscription
- dms:DeleteReplicationInstance
- dms:DeleteReplicationSubnetGroup
- dms:DeleteReplicationTask
- dms:ModifyEndpoint
- dms:ModifyEventSubscription

- `dms:ModifyReplicationInstance`
- `dms:ModifyReplicationSubnetGroup`
- `dms:ModifyReplicationTask`
- `dms:RefreshSchemas`
- `dms:RemoveTagsFromResource`
- `dms:StartReplicationTask`
- `dms:StopReplicationTask`

AWS Device Farm

These are the Write actions for AWS Device Farm.

- `devicefarm:CreateDevicePool`
- `devicefarm:CreateNetworkProfile`
- `devicefarm:CreateProject`
- `devicefarm:CreateRemoteAccessSession`
- `devicefarm:CreateUpload`
- `devicefarm:DeleteDevicePool`
- `devicefarm:DeleteNetworkProfile`
- `devicefarm:DeleteProject`
- `devicefarm:DeleteRemoteAccessSession`
- `devicefarm:DeleteRun`
- `devicefarm:DeleteUpload`
- `devicefarm:InstallToRemoteAccessSession`
- `devicefarm:PurchaseOffering`
- `devicefarm:RenewOffering`
- `devicefarm:ScheduleRun`
- `devicefarm:StopRemoteAccessSession`
- `devicefarm:StopRun`
- `devicefarm:UpdateDevicePool`
- `devicefarm:UpdateNetworkProfile`
- `devicefarm:UpdateProject`

AWS Direct Connect

These are the Write actions for AWS Direct Connect.

- `directconnect:AllocateConnectionOnInterconnect`
- `directconnect:AllocatePrivateVirtualInterface`
- `directconnect:AllocatePublicVirtualInterface`
- `directconnect>CreateConnection`
- `directconnect>CreateInterconnect`
- `directconnect>CreatePrivateVirtualInterface`
- `directconnect>CreatePublicVirtualInterface`
- `directconnect>DeleteConnection`
- `directconnect>DeleteInterconnect`
- `directconnect>DeleteVirtualInterface`

AWS Directory Service

These are the Write actions for AWS Directory Service.

- `ds:AddIpRoutes`
- `ds:AddTagsToResource`
- `ds:AuthorizeApplication`
- `ds:CancelSchemaExtension`
- `ds:ConnectDirectory`
- `ds>CreateAlias`
- `ds:CreateComputer`
- `ds:CreateConditionalForwarder`
- `ds:CreateDirectory`
- `ds:CreateMicrosoftAD`
- `ds:CreateSnapshot`
- `ds:CreateTrust`
- `ds>DeleteConditionalForwarder`
- `ds:DeleteDirectory`
- `ds:DeleteSnapshot`
- `ds:DeleteTrust`
- `ds:DeregisterEventTopic`
- `ds:DisableRadius`
- `ds:DisableSso`
- `ds:EnableRadius`
- `ds:EnableSso`
- `ds:RegisterEventTopic`
- `ds:RemoveIpRoutes`
- `ds:RemoveTagsFromResource`
- `ds:RestoreFromSnapshot`
- `ds:StartSchemaExtension`
- `ds:UnauthorizeApplication`
- `ds:UpdateConditionalForwarder`
- `ds:UpdateRadius`

Amazon DynamoDB

These are the Write actions for Amazon DynamoDB.

- `dynamodb:BatchWriteItem`
- `dynamodb CreateTable`
- `dynamodb>DeleteItem`
- `dynamodb>DeleteTable`
- `dynamodb:PurchaseReservedCapacityOfferings`
- `dynamodb:PutItem`
- `dynamodb:TagResource`
- `dynamodb:UntagResource`
- `dynamodb:UpdateItem`

- dynamodb:UpdateTable

Amazon DynamoDB Accelerator (DAX)

These are the Write actions for Amazon DynamoDB Accelerator (DAX).

- dax:BatchWriteItem
- dax>CreateCluster
- dax>CreateParameterGroup
- dax>CreateSubnetGroup
- dax:DecreaseReplicationFactor
- dax>DeleteCluster
- dax>DeleteItem
- dax>DeleteParameterGroup
- dax>DeleteSubnetGroup
- dax:IncreaseReplicationFactor
- dax:PutItem
- dax:RebootNode
- dax:TagResource
- dax:UntagResource
- dax:UpdateCluster
- dax:UpdateItem
- dax:UpdateParameterGroup
- dax:UpdateSubnetGroup

Amazon EC2

These are the Write actions for Amazon EC2.

- ec2:AcceptReservedInstancesExchangeQuote
- ec2:AcceptVpcPeeringConnection
- ec2:AllocateAddress
- ec2:AllocateHosts
- ec2:AssignIpv6Addresses
- ec2:AssignPrivateIpAddresses
- ec2:AssociateAddress
- ec2:AssociateDhcpOptions
- ec2:AssociateIamInstanceProfile
- ec2:AssociateRouteTable
- ec2:AssociateSubnetCidrBlock
- ec2:AssociateVpcCidrBlock
- ec2:AttachClassicLinkVpc
- ec2:AttachInternetGateway
- ec2:AttachNetworkInterface
- ec2:AttachVolume
- ec2:AttachVpnGateway
- ec2:AuthorizeSecurityGroupEgress

- `ec2:AuthorizeSecurityGroupIngress`
- `ec2:BundleInstance`
- `ec2:CancelBundleTask`
- `ec2:CancelConversionTask`
- `ec2:CancelExportTask`
- `ec2:CancelImportTask`
- `ec2:CancelReservedInstancesListing`
- `ec2:CancelSpotFleetRequests`
- `ec2:CancelSpotInstanceRequests`
- `ec2:ConfirmProductInstance`
- `ec2:CopyImage`
- `ec2:CopySnapshot`
- `ec2>CreateCustomerGateway`
- `ec2:CreateDhcpOptions`
- `ec2:CreateFlowLogs`
- `ec2:CreateFpgaImage`
- `ec2:CreateImage`
- `ec2:CreateInstanceExportTask`
- `ec2:CreateInternetGateway`
- `ec2:CreateKeyPair`
- `ec2:CreateNatGateway`
- `ec2:CreateNetworkAcl`
- `ec2:CreateNetworkAclEntry`
- `ec2:CreateNetworkInterface`
- `ec2:CreateNetworkInterfacePermission`
- `ec2:CreatePlacementGroup`
- `ec2:CreateReservedInstancesListing`
- `ec2:CreateRoute`
- `ec2:CreateRouteTable`
- `ec2:CreateSecurityGroup`
- `ec2:CreateSnapshot`
- `ec2:CreateSpotDatafeedSubscription`
- `ec2:CreateSubnet`
- `ec2:CreateTags`
- `ec2:CreateVolume`
- `ec2:CreateVpc`
- `ec2:CreateVpcEndpoint`
- `ec2:CreateVpcPeeringConnection`
- `ec2:CreateVpnConnection`
- `ec2:CreateVpnConnectionRoute`
- `ec2:CreateVpnGateway`
- `ec2>DeleteCustomerGateway`
- `ec2>DeleteDhcpOptions`
- `ec2>DeleteFlowLogs`
- `ec2>DeleteInternetGateway`
- `ec2>DeleteKeyPair`

- `ec2:DeleteNatGateway`
- `ec2:DeleteNetworkAcl`
- `ec2:DeleteNetworkAclEntry`
- `ec2:DeleteNetworkInterface`
- `ec2:DeletePlacementGroup`
- `ec2:DeleteRoute`
- `ec2:DeleteRouteTable`
- `ec2:DeleteSecurityGroup`
- `ec2:DeleteSnapshot`
- `ec2:DeleteSpotDatafeedSubscription`
- `ec2:DeleteSubnet`
- `ec2:DeleteTags`
- `ec2:DeleteVolume`
- `ec2:DeleteVpc`
- `ec2:DeleteVpcEndpoints`
- `ec2:DeleteVpcPeeringConnection`
- `ec2:DeleteVpnConnection`
- `ec2:DeleteVpnConnectionRoute`
- `ec2:DeleteVpnGateway`
- `ec2:DeregisterImage`
- `ec2:DescribeFpgaImages`
- `ec2:DetachClassicLinkVpc`
- `ec2:DetachInternetGateway`
- `ec2:DetachNetworkInterface`
- `ec2:DetachVolume`
- `ec2:DetachVpnGateway`
- `ec2:DisableVgwRoutePropagation`
- `ec2:DisableVpcClassicLink`
- `ec2:DisableVpcClassicLinkDnsSupport`
- `ec2:DisassociateAddress`
- `ec2:DisassociateIamInstanceProfile`
- `ec2:DisassociateRouteTable`
- `ec2:DisassociateSubnetCidrBlock`
- `ec2:DisassociateVpcCidrBlock`
- `ec2:EnableVgwRoutePropagation`
- `ec2:EnableVolumeIO`
- `ec2:EnableVpcClassicLink`
- `ec2:EnableVpcClassicLinkDnsSupport`
- `ec2:ImportImage`
- `ec2:ImportInstance`
- `ec2:ImportKeyPair`
- `ec2:ImportSnapshot`
- `ec2:ImportVolume`
- `ec2:ModifyHosts`
- `ec2:ModifyIdFormat`
- `ec2:ModifyIdentityIdFormat`

- `ec2:ModifyImageAttribute`
- `ec2:ModifyInstanceAttribute`
- `ec2:ModifyInstancePlacement`
- `ec2:ModifyNetworkInterfaceAttribute`
- `ec2:ModifyReservedInstances`
- `ec2:ModifySnapshotAttribute`
- `ec2:ModifySpotFleetRequest`
- `ec2:ModifySubnetAttribute`
- `ec2:ModifyVolume`
- `ec2:ModifyVolumeAttribute`
- `ec2:ModifyVpcAttribute`
- `ec2:ModifyVpcEndpoint`
- `ec2:ModifyVpcPeeringConnectionOptions`
- `ec2:MonitorInstances`
- `ec2:MoveAddressToVpc`
- `ec2:PurchaseHostReservation`
- `ec2:PurchaseReservedInstancesOffering`
- `ec2:PurchaseScheduledInstances`
- `ec2:RebootInstances`
- `ec2:RegisterImage`
- `ec2:RejectVpcPeeringConnection`
- `ec2:ReleaseAddress`
- `ec2:ReleaseHosts`
- `ec2:ReplaceIamInstanceProfileAssociation`
- `ec2:ReplaceNetworkAclAssociation`
- `ec2:ReplaceNetworkAclEntry`
- `ec2:ReplaceRoute`
- `ec2:ReplaceRouteTableAssociation`
- `ec2:ReportInstanceStatus`
- `ec2:RequestSpotFleet`
- `ec2:RequestSpotInstances`
- `ec2:ResetImageAttribute`
- `ec2:ResetInstanceAttribute`
- `ec2:ResetNetworkInterfaceAttribute`
- `ec2:ResetSnapshotAttribute`
- `ec2:RestoreAddressToClassic`
- `ec2:RevokeSecurityGroupEgress`
- `ec2:RevokeSecurityGroupIngress`
- `ec2:RunInstances`
- `ec2:RunScheduledInstances`
- `ec2:StartInstances`
- `ec2:StopInstances`
- `ec2:TerminateInstances`
- `ec2:UnassignPrivateIpAddresses`
- `ec2:UnmonitorInstances`
- `ec2:UpdateSecurityGroupRuleDescriptionsEgress`

- `ec2:UpdateSecurityGroupRuleDescriptionsIngress`

Amazon EC2 Container Registry

These are the Write actions for Amazon EC2 Container Registry.

- `ecr:BatchDeleteImage`
- `ecr:CompleteLayerUpload`
- `ecr>CreateRepository`
- `ecr>DeleteRepository`
- `ecr>DeleteRepositoryPolicy`
- `ecr:InitiateLayerUpload`
- `ecr:PutImage`
- `ecr:SetRepositoryPolicy`
- `ecr:UploadLayerPart`

Amazon EC2 Container Service

These are the Write actions for Amazon EC2 Container Service.

- `ecs>CreateCluster`
- `ecs>CreateService`
- `ecs>DeleteCluster`
- `ecs>DeleteService`
- `ecs:DeregisterContainerInstance`
- `ecs:DeregisterTaskDefinition`
- `ecs:DiscoverPollEndpoint`
- `ecs:Poll`
- `ecs:RegisterContainerInstance`
- `ecs:RegisterTaskDefinition`
- `ecs:RunTask`
- `ecs:StartTask`
- `ecs:StartTelemetrySession`
- `ecs:StopTask`
- `ecs:SubmitContainerStateChange`
- `ecs:SubmitTaskStateChange`
- `ecs:UpdateContainerAgent`
- `ecs:UpdateService`

Amazon ElastiCache

These are the Write actions for Amazon ElastiCache.

- `elasticache:AddTagsToResource`
- `elasticache:AuthorizeCacheSecurityGroupIngress`
- `elasticache:CopySnapshot`
- `elasticache>CreateCacheCluster`
- `elasticache>CreateCacheParameterGroup`

- elasticache:CreateCacheSecurityGroup
- elasticache:CreateCacheSubnetGroup
- elasticache:CreateReplicationGroup
- elasticache:CreateSnapshot
- elasticache:DeleteCacheCluster
- elasticache:DeleteCacheParameterGroup
- elasticache:DeleteCacheSecurityGroup
- elasticache:DeleteCacheSubnetGroup
- elasticache:DeleteReplicationGroup
- elasticache:DeleteSnapshot
- elasticache:ModifyCacheCluster
- elasticache:ModifyCacheParameterGroup
- elasticache:ModifyCacheSubnetGroup
- elasticache:ModifyReplicationGroup
- elasticache:PurchaseReservedCacheNodesOffering
- elasticache:RebootCacheCluster
- elasticache:RemoveTagsFromResource
- elasticache:ResetCacheParameterGroup
- elasticache:RevokeCacheSecurityGroupIngress

AWS Elastic Beanstalk

These are the Write actions for AWS Elastic Beanstalk.

- elasticbeanstalk:AbortEnvironmentUpdate
- elasticbeanstalk:ApplyEnvironmentManagedAction
- elasticbeanstalk:ComposeEnvironments
- elasticbeanstalk>CreateApplication
- elasticbeanstalk>CreateApplicationVersion
- elasticbeanstalk>CreateConfigurationTemplate
- elasticbeanstalk>CreateEnvironment
- elasticbeanstalk>CreatePlatformVersion
- elasticbeanstalk>CreateStorageLocation
- elasticbeanstalk>DeleteApplication
- elasticbeanstalk>DeleteApplicationVersion
- elasticbeanstalk>DeleteConfigurationTemplate
- elasticbeanstalk>DeleteEnvironmentConfiguration
- elasticbeanstalk>DeletePlatformVersion
- elasticbeanstalk:RebuildEnvironment
- elasticbeanstalk:RestartAppServer
- elasticbeanstalk:SwapEnvironmentCNAMEs
- elasticbeanstalk:TerminateEnvironment
- elasticbeanstalk:UpdateApplication
- elasticbeanstalk:UpdateApplicationResourceLifecycle
- elasticbeanstalk:UpdateApplicationVersion
- elasticbeanstalk:UpdateConfigurationTemplate

- elasticbeanstalk:UpdateEnvironment

Amazon Elastic File System

These are the Write actions for Amazon Elastic File System.

- elasticfilesystem>CreateFileSystem
- elasticfilesystem>CreateMountTarget
- elasticfilesystem>CreateTags
- elasticfilesystem>DeleteFileSystem
- elasticfilesystem>DeleteMountTarget
- elasticfilesystem>DeleteTags
- elasticfilesystem>ModifyMountTargetSecurityGroups

Elastic Load Balancing V2

These are the Write actions for Elastic Load Balancing V2.

- elasticloadbalancing>AddTags
- elasticloadbalancing>CreateListener
- elasticloadbalancing>CreateLoadBalancer
- elasticloadbalancing>CreateRule
- elasticloadbalancing>CreateTargetGroup
- elasticloadbalancing>DeleteListener
- elasticloadbalancing>DeleteLoadBalancer
- elasticloadbalancing>DeleteRule
- elasticloadbalancing>DeleteTargetGroup
- elasticloadbalancing>DeregisterTargets
- elasticloadbalancing>ModifyListener
- elasticloadbalancing>ModifyLoadBalancerAttributes
- elasticloadbalancing>ModifyRule
- elasticloadbalancing>ModifyTargetGroup
- elasticloadbalancing>ModifyTargetGroupAttributes
- elasticloadbalancing>RegisterTargets
- elasticloadbalancing>RemoveTags
- elasticloadbalancing>SetIpAddressType
- elasticloadbalancing>SetRulePriorities
- elasticloadbalancing>SetSecurityGroups
- elasticloadbalancing>SetSubnets

Amazon Elastic MapReduce

These are the Write actions for Amazon Elastic MapReduce.

- elasticmapreduce>AddInstanceGroups
- elasticmapreduce>AddJobFlowSteps
- elasticmapreduce>AddTags
- elasticmapreduce>CancelSteps

- elasticmapreduce:CreateSecurityConfiguration
- elasticmapreduce:DeleteSecurityConfiguration
- elasticmapreduce:ModifyInstanceGroups
- elasticmapreduce:PutAutoScalingPolicy
- elasticmapreduce:RemoveAutoScalingPolicy
- elasticmapreduce:RemoveTags
- elasticmapreduce:RunJobFlow
- elasticmapreduce:SetTerminationProtection
- elasticmapreduce:SetVisibleToAllUsers
- elasticmapreduce:TerminateJobFlows

Amazon Elastic Transcoder

These are the Write actions for Amazon Elastic Transcoder.

- elastictranscoder:CancelJob
- elastictranscoder>CreateJob
- elastictranscoder>CreatePipeline
- elastictranscoder>CreatePreset
- elastictranscoder>DeletePipeline
- elastictranscoder>DeletePreset
- elastictranscoder:TestRole
- elastictranscoder:UpdatePipeline
- elastictranscoder:UpdatePipelineNotifications
- elastictranscoder:UpdatePipelineStatus

Amazon Elasticsearch Service

These are the Write actions for Amazon Elasticsearch Service.

- es:AddTags
- es>CreateElasticsearchDomain
- es>DeleteElasticsearchDomain
- es:ESHttpPost
- es:ESHttpPut
- es:RemoveTags
- es:UpdateElasticsearchDomainConfig

Amazon GameLift

These are the Write actions for Amazon GameLift.

- gamelift>CreateAlias
- gamelift>CreateBuild
- gamelift>CreateFleet
- gamelift>CreateGameSession
- gamelift>CreatePlayerSession
- gamelift>CreatePlayerSessions

- gamelift:DeleteAlias
- gamelift:DeleteBuild
- gamelift:DeleteFleet
- gamelift:DeleteScalingPolicy
- gamelift:PutScalingPolicy
- gamelift:UpdateAlias
- gamelift:UpdateBuild
- gamelift:UpdateFleetAttributes
- gamelift:UpdateFleetCapacity
- gamelift:UpdateFleetPortSettings
- gamelift:UpdateGameSession
- gamelift:UpdateRuntimeConfiguration

Amazon Glacier

These are the Write actions for Amazon Glacier.

- glacier:AbortMultipartUpload
- glacier:AddTagsToVault
- glacier:CompleteMultipartUpload
- glacier>CreateVault
- glacier>DeleteArchive
- glacier>DeleteVault
- glacier>DeleteVaultNotifications
- glacier:InitiateJob
- glacier:InitiateMultipartUpload
- glacier:PurchaseProvisionedCapacity
- glacier:RemoveTagsFromVault
- glacier:SetVaultNotifications
- glacier:UploadArchive
- glacier:UploadMultipartPart

Identity And Access Management

These are the Write actions for Identity And Access Management.

- iam:AddClientIDToOpenIDConnectProvider
- iam:AddRoleToInstanceProfile
- iam:AddUserToGroup
- iam:ChangePassword
- iam>CreateAccessKey
- iam>CreateAccountAlias
- iam>CreateGroup
- iam>CreateInstanceProfile
- iam>CreateLoginProfile
- iam>CreateOpenIDConnectProvider
- iam>CreateRole

- iam:CreateSAMLProvider
- iam:CreateServiceSpecificCredential
- iam:CreateUser
- iam:CreateVirtualMFADevice
- iam:DeactivateMFADevice
- iam:DeleteAccessKey
- iam:DeleteAccountAlias
- iam:DeleteGroup
- iam:DeleteInstanceProfile
- iam:DeleteLoginProfile
- iam:DeleteOpenIDConnectProvider
- iam:DeleteRole
- iam:DeleteSAMLProvider
- iam:DeleteSSHPublicKey
- iam:DeleteServerCertificate
- iam:DeleteServiceSpecificCredential
- iam:DeleteSigningCertificate
- iam:DeleteUser
- iam:DeleteVirtualMFADevice
- iam:EnableMFADevice
- iam:PassRole
- iam:RemoveClientIDFromOpenIDConnectProvider
- iam:RemoveRoleFromInstanceProfile
- iam:RemoveUserFromGroup
- iam:ResetServiceSpecificCredential
- iam:ResyncMFADevice
- iam:UpdateAccessKey
- iam:UpdateAccountPasswordPolicy
- iam:UpdateGroup
- iam:UpdateLoginProfile
- iam:UpdateOpenIDConnectProviderThumbprint
- iam:UpdateSAMLProvider
- iam:UpdateSSHPublicKey
- iam:UpdateServerCertificate
- iam:UpdateServiceSpecificCredential
- iam:UpdateSigningCertificate
- iam:UpdateUser
- iam:UploadSSHPublicKey
- iam:UploadServerCertificate
- iam:UploadSigningCertificate

AWS Import Export Disk Service

These are the Write actions for AWS Import Export Disk Service.

- importexport:CancelJob

- `importexport:CreateJob`
- `importexport:UpdateJob`

Amazon Inspector

These are the Write actions for Amazon Inspector.

- `inspector:AddAttributesToFindings`
- `inspector>CreateAssessmentTarget`
- `inspector>CreateAssessmentTemplate`
- `inspector>CreateResourceGroup`
- `inspector>DeleteAssessmentRun`
- `inspector>DeleteAssessmentTarget`
- `inspector>DeleteAssessmentTemplate`
- `inspector:RegisterCrossAccountAccessRole`
- `inspector>RemoveAttributesFromFindings`
- `inspector>SetTagsForResource`
- `inspector>StartAssessmentRun`
- `inspector>StopAssessmentRun`
- `inspector>SubscribeToEvent`
- `inspector>UnsubscribeFromEvent`
- `inspector>UpdateAssessmentTarget`

AWS IoT

These are the Write actions for AWS IoT.

- `iot:AcceptCertificateTransfer`
- `iot:AttachThingPrincipal`
- `iot:CancelCertificateTransfer`
- `iot:Connect`
- `iot>CreateCertificateFromCsr`
- `iot>CreateKeysAndCertificate`
- `iot>CreateThing`
- `iot>CreateThingType`
- `iot>CreateTopicRule`
- `iot>DeleteCACertificate`
- `iot>DeleteCertificate`
- `iot>DeleteRegistrationCode`
- `iot>DeleteThing`
- `iot>DeleteThingShadow`
- `iot>DeleteThingType`
- `iot>DeleteTopicRule`
- `iot>DeprecateThingType`
- `iot:DetachThingPrincipal`
- `iot:DisableTopicRule`
- `iot:EnableTopicRule`

- iot:Publish
- iot:Receive
- iot:RegisterCACertificate
- iot:RegisterCertificate
- iot:RejectCertificateTransfer
- iot:ReplaceTopicRule
- iot:SetLoggingOptions
- iot:Subscribe
- iot:TransferCertificate
- iot:UpdateCACertificate
- iot:UpdateCertificate
- iot:UpdateThing
- iot:UpdateThingShadow

AWS Key Management Service

These are the Write actions for AWS Key Management Service.

- kms:CancelKeyDeletion
- kms>CreateAlias
- kms:Decrypt
- kms>DeleteAlias
- kms>DeleteImportedKeyMaterial
- kms:DisableKey
- kms:DisableKeyRotation
- kms:EnableKey
- kms:EnableKeyRotation
- kms:Encrypt
- kms:GenerateDataKey
- kms:GenerateDataKeyWithoutPlaintext
- kms:ImportKeyMaterial
- kms:ScheduleKeyDeletion
- kms:TagResource
- kms:UntagResource
- kms:UpdateAlias
- kms:UpdateKeyDescription

Amazon Kinesis

These are the Write actions for Amazon Kinesis.

- kinesis:AddTagsToStream
- kinesis>CreateStream
- kinesis:DecreaseStreamRetentionPeriod
- kinesis>DeleteStream
- kinesis:DisableEnhancedMonitoring
- kinesis:EnableEnhancedMonitoring

- `kinesis:IncreaseStreamRetentionPeriod`
- `kinesis:MergeShards`
- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:RemoveTagsFromStream`
- `kinesis:SplitShard`
- `kinesis:UpdateShardCount`

Amazon Kinesis Analytics

These are the Write actions for Amazon Kinesis Analytics.

- `kinesisanalytics:AddApplicationInput`
- `kinesisanalytics:AddApplicationOutput`
- `kinesisanalytics:AddApplicationReferenceDataSource`
- `kinesisanalytics>CreateApplication`
- `kinesisanalytics>DeleteApplication`
- `kinesisanalytics>DeleteApplicationOutput`
- `kinesisanalytics>DeleteApplicationReferenceDataSource`
- `kinesisanalytics:StartApplication`
- `kinesisanalytics:StopApplication`
- `kinesisanalytics:UpdateApplication`

Amazon Kinesis Firehose

These are the Write actions for Amazon Kinesis Firehose.

- `firehose:CreateDeliveryStream`
- `firehose:DeleteDeliveryStream`
- `firehose:PutRecord`
- `firehose:PutRecordBatch`
- `firehose:UpdateDestination`

AWS Lambda

These are the Write actions for AWS Lambda.

- `lambda>CreateAlias`
- `lambda>CreateEventSourceMapping`
- `lambda>CreateFunction`
- `lambda>DeleteAlias`
- `lambda>DeleteEventSourceMapping`
- `lambda>DeleteFunction`
- `lambda:Invoke`
- `lambda:InvokeAsync`
- `lambda:InvokeFunction`
- `lambda:PublishVersion`
- `lambda:TagResource`

- `lambda:UntagResource`
- `lambda:UpdateAlias`
- `lambda:UpdateEventSourceMapping`
- `lambda:UpdateFunctionCode`
- `lambda:UpdateFunctionConfiguration`

Amazon Lex

These are the Write actions for Amazon Lex.

- `lex:CreateBotVersion`
- `lex:CreateIntentVersion`
- `lex:CreateSlotTypeVersion`
- `lex:DeleteBot`
- `lex:DeleteBotAlias`
- `lex:DeleteBotChannelAssociation`
- `lex:DeleteBotVersion`
- `lex:DeleteIntent`
- `lex:DeleteIntentVersion`
- `lex:DeleteSlotType`
- `lex:DeleteSlotTypeVersion`
- `lex:DeleteUtterances`
- `lex:PostContent`
- `lex:PostText`
- `lex:PutBot`
- `lex:PutBotAlias`
- `lex:PutIntent`
- `lex:PutSlotType`

Amazon Lightsail

These are the Write actions for Amazon Lightsail.

- `lightsail:AllocateStaticIp`
- `lightsail:AttachStaticIp`
- `lightsail:CloseInstancePublicPorts`
- `lightsail>CreateDomain`
- `lightsail>CreateDomainEntry`
- `lightsail>CreateInstanceSnapshot`
- `lightsail>CreateInstances`
- `lightsail>CreateInstancesFromSnapshot`
- `lightsail>CreateKeyPair`
- `lightsail>DeleteDomain`
- `lightsail>DeleteDomainEntry`
- `lightsail>DeleteInstance`
- `lightsail>DeleteInstanceSnapshot`
- `lightsail>DeleteKeyPair`

- lightsail:DetachStaticIp
- lightsail:ImportKeyPair
- lightsail:OpenInstancePublicPorts
- lightsail:PeerVpc
- lightsail:RebootInstance
- lightsail:ReleaseStaticIp
- lightsail:StartInstance
- lightsail:StopInstance
- lightsail:UnpeerVpc
- lightsail:UpdateDomainEntry

Amazon Machine Learning

These are the Write actions for Amazon Machine Learning.

- machinelearning:AddTags
- machinelearning>CreateBatchPrediction
- machinelearning>CreateDataSourceFromRDS
- machinelearning>CreateDataSourceFromRedshift
- machinelearning>CreateDataSourceFromS3
- machinelearning>CreateEvaluation
- machinelearning>CreateMLModel
- machinelearning>CreateRealtimeEndpoint
- machinelearning>DeleteBatchPrediction
- machinelearning>DeleteDataSource
- machinelearning>DeleteEvaluation
- machinelearning>DeleteMLModel
- machinelearning>DeleteRealtimeEndpoint
- machinelearning>DeleteTags
- machinelearning>Predict
- machinelearning>UpdateBatchPrediction
- machinelearning>UpdateDataSource
- machinelearning>UpdateEvaluation
- machinelearning>UpdateMLModel

Manage Amazon API Gateway

These are the Write actions for Manage Amazon API Gateway.

- apigateway:DELETE
- apigateway:PATCH
- apigateway:POST
- apigateway:PUT

AWS Marketplace

These are the Write actions for AWS Marketplace.

- aws-marketplace:Subscribe
- aws-marketplace:Unsubscribe

AWS Marketplace Management Portal

These are the Write actions for AWS Marketplace Management Portal.

- aws-marketplace-management:uploadFiles

Amazon Mechanical Turk

These are the Write actions for Amazon Mechanical Turk.

- mechanicturk:ApproveAssignment
- mechanicturk:ApproveRejectedAssignment
- mechanicturk:AssignQualification
- mechanicturk:BlockWorker
- mechanicturk:ChangeHITTypeOfHIT
- mechanicturk>CreateHIT
- mechanicturk>CreateQualificationType
- mechanicturk:DisableHIT
- mechanicturk:DisposeHIT
- mechanicturk:DisposeQualificationType
- mechanicturk:ExtendHIT
- mechanicturk:ForceExpireHIT
- mechanicturk:GrantBonus
- mechanicturk:GrantQualification
- mechanicturk:NotifyWorkers
- mechanicturk:RegisterHITType
- mechanicturk:RejectAssignment
- mechanicturk:RejectQualificationRequest
- mechanicturk:RevokeQualification
- mechanicturk:SendTestEventNotification
- mechanicturk:SetHITAsReviewing
- mechanicturk:SetHITTypeNotification
- mechanicturk:UnblockWorker
- mechanicturk:UpdateQualificationScore
- mechanicturk:UpdateQualificationType

Amazon Message Delivery Service

These are the Write actions for Amazon Message Delivery Service.

- ec2messages:AcknowledgeMessage
- ec2messages:DeleteMessage
- ec2messages:FailMessage
- ec2messages:SendReply

Amazon Mobile Analytics

These are the Write actions for Amazon Mobile Analytics.

- `mobileanalytics:PutEvents`

AWS Mobile Hub

These are the Write actions for AWS Mobile Hub.

- `mobilehub>CreateProject`
- `mobilehub>CreateServiceRole`
- `mobilehub>DeleteProject`
- `mobilehub:DeployToStage`
- `mobilehub:GenerateProjectParameters`
- `mobilehub:SynchronizeProject`
- `mobilehub:UpdateProject`

AWS OpsWorks

These are the Write actions for AWS OpsWorks.

- `opsworks:AssignInstance`
- `opsworks:AssignVolume`
- `opsworks:AssociateElasticIp`
- `opsworks:AttachElasticLoadBalancer`
- `opsworks:CloneStack`
- `opsworks>CreateApp`
- `opsworks>CreateDeployment`
- `opsworks>CreateInstance`
- `opsworks>CreateLayer`
- `opsworks>CreateStack`
- `opsworks>CreateUserProfile`
- `opsworks>DeleteApp`
- `opsworks>DeleteInstance`
- `opsworks>DeleteLayer`
- `opsworks>DeleteStack`
- `opsworks>DeleteUserProfile`
- `opsworks:DeregisterEcsCluster`
- `opsworks:DeregisterElasticIp`
- `opsworks:DeregisterInstance`
- `opsworks:DeregisterRdsDbInstance`
- `opsworks:DeregisterVolume`
- `opsworks:DetachElasticLoadBalancer`
- `opsworks:DisassociateElasticIp`
- `opsworks:GrantAccess`
- `opsworks:RebootInstance`
- `opsworks:RegisterEcsCluster`

- opsworks:RegisterElasticIp
- opsworks:RegisterInstance
- opsworks:RegisterRdsDbInstance
- opsworks:RegisterVolume
- opsworks:SetLoadBasedAutoScaling
- opsworks:SetTimeBasedAutoScaling
- opsworks:StartInstance
- opsworks:StartStack
- opsworks:StopInstance
- opsworks:StopStack
- opsworks:UnassignInstance
- opsworks:UnassignVolume
- opsworks:UpdateApp
- opsworks:UpdateElasticIp
- opsworks:UpdateInstance
- opsworks:UpdateLayer
- opsworks:UpdateMyUserProfile
- opsworks:UpdateRdsDbInstance
- opsworks:UpdateStack
- opsworks:UpdateVolume

AWS OpsWorks Configuration Management

These are the Write actions for AWS OpsWorks Configuration Management.

- opsworks-cm:AssociateNode
- opsworks-cm>CreateBackup
- opsworks-cm>CreateServer
- opsworks-cm>DeleteBackup
- opsworks-cm>DeleteServer
- opsworks-cm:DisassociateNode
- opsworks-cm:RestoreServer
- opsworks-cm:StartMaintenance
- opsworks-cm:UpdateServer
- opsworks-cm:UpdateServerEngineAttributes

AWS Organizations

These are the Write actions for AWS Organizations.

- organizations:AcceptHandshake
- organizations:AttachPolicy
- organizations:CancelHandshake
- organizations>CreateAccount
- organizations>CreateOrganization
- organizations>CreateOrganizationalUnit
- organizations>CreatePolicy

- organizations:DeclineHandshake
- organizations:DeleteOrganization
- organizations:DeleteOrganizationalUnit
- organizations:DeletePolicy
- organizations:DetachPolicy
- organizations:DisablePolicyType
- organizations:EnableFullControl
- organizations:EnablePolicyType
- organizations:InviteAccountToOrganization
- organizations:LeaveOrganization
- organizations:MoveAccount
- organizations:RemoveAccountFromOrganization
- organizations:UpdateOrganizationalUnit
- organizations:UpdatePolicy

Amazon Pinpoint

These are the Write actions for Amazon Pinpoint.

- mobiletargeting>CreateCampaign
- mobiletargeting>CreateImportJob
- mobiletargeting>CreateSegment
- mobiletargeting>DeleteApnsChannel
- mobiletargeting>DeleteCampaign
- mobiletargeting>DeleteGcmChannel
- mobiletargeting>DeleteSegment
- mobiletargeting>UpdateApnsChannel
- mobiletargeting>UpdateApplicationSettings
- mobiletargeting>UpdateCampaign
- mobiletargeting>UpdateEndpoint
- mobiletargeting>UpdateEndpointsBatch
- mobiletargeting>UpdateGcmChannel
- mobiletargeting>UpdateSegment

Amazon Polly

These are the Write actions for Amazon Polly.

- polly:DeleteLexicon
- polly:PutLexicon

Amazon RDS

These are the Write actions for Amazon RDS.

- rds:AddRoleToDBCluster
- rds:AddSourceIdentifierToSubscription
- rds:AddTagsToResource

- rds:ApplyPendingMaintenanceAction
- rds:CopyDBClusterSnapshot
- rds:CopyDBParameterGroup
- rds:CopyDBSnapshot
- rds:CopyOptionGroup
- rds>CreateDBCluster
- rds>CreateDBClusterParameterGroup
- rds>CreateDBClusterSnapshot
- rds>CreateDBInstance
- rds>CreateDBInstanceReadReplica
- rds>CreateDBParameterGroup
- rds>CreateDBSecurityGroup
- rds>CreateDBSnapshot
- rds>CreateDBSubnetGroup
- rds>CreateEventSubscription
- rds>CreateOptionGroup
- rds>DeleteDBCluster
- rds>DeleteDBClusterParameterGroup
- rds>DeleteDBClusterSnapshot
- rds>DeleteDBInstance
- rds>DeleteDBParameterGroup
- rds>DeleteDBSecurityGroup
- rds>DeleteDBSnapshot
- rds>DeleteDBSubnetGroup
- rds>DeleteEventSubscription
- rds>DeleteOptionGroup
- rds:FailoverDBCluster
- rds:ModifyDBCluster
- rds:ModifyDBClusterParameterGroup
- rds:ModifyDBClusterSnapshotAttribute
- rds:ModifyDBInstance
- rds:ModifyDBParameterGroup
- rds:ModifyDBSnapshotAttribute
- rds:ModifyDBSubnetGroup
- rds:ModifyEventSubscription
- rds:ModifyOptionGroup
- rds>PromoteReadReplica
- rds>PurchaseReservedDBInstancesOffering
- rds:RebootDBInstance
- rds:RemoveSourceIdentifierFromSubscription
- rds:RemoveTagsFromResource
- rds:ResetDBClusterParameterGroup
- rds:ResetDBParameterGroup
- rds:RestoreDBClusterFromSnapshot
- rds:RestoreDBClusterToPointInTime

- `rds:RestoreDBInstanceFromDBSnapshot`
- `rds:RestoreDBInstanceToPointInTime`
- `rds:RevokeDBSecurityGroupIngress`
- `rds:StartDBInstance`
- `rds:StopDBInstance`

Amazon Redshift

These are the Write actions for Amazon Redshift.

- `redshift:AuthorizeClusterSecurityGroupIngress`
- `redshift:CancelQuerySession`
- `redshift:CopyClusterSnapshot`
- `redshift>CreateCluster`
- `redshift:CreateClusterParameterGroup`
- `redshift:CreateClusterSecurityGroup`
- `redshift:CreateClusterSnapshot`
- `redshift:CreateClusterSubnetGroup`
- `redshift:CreateEventSubscription`
- `redshift:CreateHsmClientCertificate`
- `redshift:CreateHsmConfiguration`
- `redshift:CreateTags`
- `redshift>DeleteCluster`
- `redshift:DeleteClusterParameterGroup`
- `redshift:DeleteClusterSecurityGroup`
- `redshift:DeleteClusterSnapshot`
- `redshift:DeleteClusterSubnetGroup`
- `redshift:DeleteEventSubscription`
- `redshift:DeleteHsmClientCertificate`
- `redshift:DeleteHsmConfiguration`
- `redshift:DeleteSnapshotCopyGrant`
- `redshift:DeleteTags`
- `redshift:DisableLogging`
- `redshift:DisableSnapshotCopy`
- `redshift:EnableLogging`
- `redshift:EnableSnapshotCopy`
- `redshift:ModifyCluster`
- `redshift:ModifyClusterParameterGroup`
- `redshift:ModifyClusterSubnetGroup`
- `redshift:ModifyEventSubscription`
- `redshift:ModifySnapshotCopyRetentionPeriod`
- `redshift:PurchaseReservedNodeOffering`
- `redshift:RebootCluster`
- `redshift:ResetClusterParameterGroup`
- `redshift:RestoreFromClusterSnapshot`
- `redshift:RestoreTableFromClusterSnapshot`

Amazon Rekognition

These are the Write actions for Amazon Rekognition.

- `rekognition:CreateCollection`
- `rekognition:DeleteCollection`
- `rekognition:DeleteFaces`
- `rekognition:IndexFaces`

Amazon Resource Group Tagging API

These are the Write actions for Amazon Resource Group Tagging API.

- `tag:AddResourceTags`
- `tag:RemoveResourceTags`
- `tag:TagResources`
- `tag:UntagResources`

Amazon Route 53

These are the Write actions for Amazon Route 53.

- `route53:AssociateVPCWithHostedZone`
- `route53:ChangeResourceRecordSets`
- `route53:ChangeTagsForResource`
- `route53:CreateHealthCheck`
- `route53:CreateHostedZone`
- `route53:CreateReusableDelegationSet`
- `route53:CreateTrafficPolicy`
- `route53:CreateTrafficPolicyInstance`
- `route53:CreateTrafficPolicyVersion`
- `route53:DeleteHealthCheck`
- `route53:DeleteHostedZone`
- `route53:DeleteReusableDelegationSet`
- `route53:DeleteTrafficPolicy`
- `route53:DeleteTrafficPolicyInstance`
- `route53:DisableDomainAutoRenew`
- `route53:DisassociateVPCFromHostedZone`
- `route53:EnableDomainAutoRenew`
- `route53:UpdateHealthCheck`
- `route53:UpdateHostedZoneComment`
- `route53:UpdateTrafficPolicyComment`
- `route53:UpdateTrafficPolicyInstance`

Amazon Route53 Domains

These are the Write actions for Amazon Route53 Domains.

- `route53domains>DeleteTagsForDomain`

- route53domains:DisableDomainAutoRenew
- route53domains:DisableDomainTransferLock
- route53domains:EnableDomainAutoRenew
- route53domains:EnableDomainTransferLock
- route53domains:RegisterDomain
- route53domains:RenewDomain
- route53domains:ResendContactReachabilityEmail
- route53domains:RetrieveDomainAuthCode
- route53domains:TransferDomain
- route53domains:UpdateDomainContact
- route53domains:UpdateDomainContactPrivacy
- route53domains:UpdateDomainNameservers
- route53domains:UpdateTagsForDomain

Amazon S3

These are the Write actions for Amazon S3.

- s3:AbortMultipartUpload
- s3:CreateBucket
- s3:DeleteBucket
- s3:DeleteBucketWebsite
- s3:DeleteObject
- s3:DeleteObjectTagging
- s3:DeleteObjectVersion
- s3:DeleteObjectVersionTagging
- s3:PutAccelerateConfiguration
- s3:PutAnalyticsConfiguration
- s3:PutBucketCORS
- s3:PutBucketLogging
- s3:PutBucketNotification
- s3:PutBucketRequestPayment
- s3:PutBucketTagging
- s3:PutBucketVersioning
- s3:PutBucketWebsite
- s3:PutInventoryConfiguration
- s3:PutIpConfiguration
- s3:PutLifecycleConfiguration
- s3:PutMetricsConfiguration
- s3:PutObject
- s3:PutObjectTagging
- s3:PutObjectVersionTagging
- s3:PutReplicationConfiguration
- s3:ReplicateDelete
- s3:ReplicateObject
- s3:ReplicateTags

- `s3:RestoreObject`

Amazon SES

These are the Write actions for Amazon SES.

- `ses:CloneReceiptRuleSet`
- `ses>CreateReceiptFilter`
- `ses:CreateReceiptRule`
- `ses:CreateReceiptRuleSet`
- `ses>DeleteIdentity`
- `ses>DeleteIdentityPolicy`
- `ses>DeleteReceiptFilter`
- `ses>DeleteReceiptRule`
- `ses>DeleteReceiptRuleSet`
- `ses>DeleteVerifiedEmailAddress`
- `ses:PutIdentityPolicy`
- `ses:ReorderReceiptRuleSet`
- `ses:SendBounce`
- `ses:SendEmail`
- `ses:SendRawEmail`
- `ses:SetActiveReceiptRuleSet`
- `ses:SetIdentityDkimEnabled`
- `ses:SetIdentityFeedbackForwardingEnabled`
- `ses:SetIdentityHeadersInNotificationsEnabled`
- `ses:SetIdentityMailFromDomain`
- `ses:SetIdentityNotificationTopic`
- `ses:SetReceiptRulePosition`
- `ses:UpdateReceiptRule`

Amazon SNS

These are the Write actions for Amazon SNS.

- `sns:ConfirmSubscription`
- `sns>CreatePlatformApplication`
- `sns>CreatePlatformEndpoint`
- `sns:CreateTopic`
- `sns>DeleteEndpoint`
- `sns>DeletePlatformApplication`
- `sns>DeleteTopic`
- `sns:OptInPhoneNumber`
- `sns:Publish`
- `sns:SetEndpointAttributes`
- `sns:SetPlatformApplicationAttributes`
- `sns:SetsMSAttributes`
- `sns:SetSubscriptionAttributes`

- `sns:SetTopicAttributes`
- `sns:Subscribe`
- `sns:Unsubscribe`

Amazon SQS

These are the Write actions for Amazon SQS.

- `sqs:ChangeMessageVisibility`
- `sqs:ChangeMessageVisibilityBatch`
- `sqs>CreateQueue`
- `sqs>DeleteMessage`
- `sqs>DeleteMessageBatch`
- `sqs>DeleteQueue`
- `sqs:PurgeQueue`
- `sqs:SendMessage`
- `sqs:SendMessageBatch`
- `sqs:SetQueueAttributes`

AWS Security Token Service

These are the Write actions for AWS Security Token Service.

- `sts:AssumeRole`
- `sts:AssumeRoleWithSAML`
- `sts:AssumeRoleWithWebIdentity`
- `sts:DecodeAuthorizationMessage`

AWS Service Catalog

These are the Write actions for AWS Service Catalog.

- `servicecatalog:AcceptPortfolioShare`
- `servicecatalog:AssociatePrincipalWithPortfolio`
- `servicecatalog:AssociateProductWithPortfolio`
- `servicecatalog>CreateConstraint`
- `servicecatalog>CreatePortfolio`
- `servicecatalog>CreateProduct`
- `servicecatalog>CreateProvisioningArtifact`
- `servicecatalog>DeleteConstraint`
- `servicecatalog>DeletePortfolio`
- `servicecatalog>DeleteProduct`
- `servicecatalog>DeleteProvisioningArtifact`
- `servicecatalog:DisassociatePrincipalFromPortfolio`
- `servicecatalog:DisassociateProductFromPortfolio`
- `servicecatalog:ProvisionProduct`
- `servicecatalog:RejectPortfolioShare`
- `servicecatalog:TerminateProvisionedProduct`

- `servicecatalog:UpdateConstraint`
- `servicecatalog:UpdatePortfolio`
- `servicecatalog:UpdateProduct`
- `servicecatalog:UpdateProvisionedProduct`
- `servicecatalog:UpdateProvisioningArtifact`

AWS Shield

These are the Write actions for AWS Shield.

- `shield>CreateProtection`
- `shield>CreateSubscription`
- `shield>DeleteProtection`
- `shield>DeleteSubscription`

Amazon Simple Systems Manager

These are the Write actions for Amazon Simple Systems Manager.

- `ssm>AddTagsToResource`
- `ssm:CancelCommand`
- `ssm>CreateActivation`
- `ssm>CreateAssociation`
- `ssm>CreateAssociationBatch`
- `ssm>CreateDocument`
- `ssm>CreateMaintenanceWindow`
- `ssm>CreatePatchBaseline`
- `ssm>DeleteActivation`
- `ssm>DeleteAssociation`
- `ssm>DeleteDocument`
- `ssm>DeleteMaintenanceWindow`
- `ssm>DeleteParameter`
- `ssm>DeletePatchBaseline`
- `ssm:DeregisterManagedInstance`
- `ssm:DeregisterPatchBaselineForPatchGroup`
- `ssm:DeregisterTargetFromMaintenanceWindow`
- `ssm:DeregisterTaskFromMaintenanceWindow`
- `ssm:ModifyDocumentPermission`
- `ssm:PutInventory`
- `ssm:PutParameter`
- `ssm:RegisterDefaultPatchBaseline`
- `ssm:RegisterPatchBaselineForPatchGroup`
- `ssm:RegisterTargetWithMaintenanceWindow`
- `ssm:RegisterTaskWithMaintenanceWindow`
- `ssm:RemoveTagsFromResource`
- `ssm:SendCommand`
- `ssm:StartAssociationsOnce`

- `ssm:UpdateAssociation`
- `ssm:UpdateAssociationStatus`
- `ssm:UpdateDocument`
- `ssm:UpdateDocumentDefaultVersion`
- `ssm:UpdateInstanceAssociationStatus`
- `ssm:UpdateInstanceInformation`
- `ssm:UpdateMaintenanceWindow`
- `ssm:UpdateManagedInstanceRole`
- `ssm:UpdatePatchBaseline`

Amazon Simple Workflow Service

These are the Write actions for Amazon Simple Workflow Service.

- `swf:CancelTimer`
- `swf:CancelWorkflowExecution`
- `swf:CompleteWorkflowExecution`
- `swf:ContinueAsNewWorkflowExecution`
- `swf:DeprecateActivityType`
- `swf:DeprecateDomain`
- `swf:DeprecateWorkflowType`
- `swf:FailWorkflowExecution`
- `swf:PollForActivityTask`
- `swf:PollForDecisionTask`
- `swf:RecordActivityTaskHeartbeat`
- `swf:RecordMarker`
- `swf:RegisterActivityType`
- `swf:RegisterDomain`
- `swf:RegisterWorkflowType`
- `swf:RequestCancelActivityTask`
- `swf:RequestCancelExternalWorkflowExecution`
- `swf:RequestCancelWorkflowExecution`
- `swf:RespondActivityTaskCanceled`
- `swf:RespondActivityTaskCompleted`
- `swf:RespondActivityTaskFailed`
- `swf:RespondDecisionTaskCompleted`
- `swf:ScheduleActivityTask`
- `swf:SignalExternalWorkflowExecution`
- `swf:SignalWorkflowExecution`
- `swf:StartChildWorkflowExecution`
- `swf:StartTimer`
- `swf:StartWorkflowExecution`
- `swf:TerminateWorkflowExecution`

Amazon SimpleDB

These are the Write actions for Amazon SimpleDB.

- `sdb:BatchDeleteAttributes`
- `sdb:BatchPutAttributes`
- `sdb>CreateDomain`
- `sdb>DeleteAttributes`
- `sdb>DeleteDomain`
- `sdb:DomainMetadata`
- `sdb:PutAttributes`

AWS Step Functions

These are the Write actions for AWS Step Functions.

- `states:CreateActivity`
- `states:CreateStateMachine`
- `states:DeleteActivity`
- `states:DeleteStateMachine`
- `states:GetActivityTask`
- `states:SendTaskFailure`
- `states:SendTaskHeartbeat`
- `states:SendTaskSuccess`
- `states:StartExecution`
- `states:StopExecution`

Amazon Storage Gateway

These are the Write actions for Amazon Storage Gateway.

- `storagegateway:ActivateGateway`
- `storagegateway:AddCache`
- `storagegateway:AddTagsToResource`
- `storagegateway:AddUploadBuffer`
- `storagegateway:AddWorkingStorage`
- `storagegateway:CancelArchival`
- `storagegateway:CancelRetrieval`
- `storagegateway>CreateCachediSCSIVolume`
- `storagegateway>CreateSnapshot`
- `storagegateway>CreateSnapshotFromVolumeRecoveryPoint`
- `storagegateway>CreateStorediSCSIVolume`
- `storagegateway>CreateTapeWithBarcode`
- `storagegateway>CreateTapes`
- `storagegateway>DeleteBandwidthRateLimit`
- `storagegateway>DeleteChapCredentials`
- `storagegateway>DeleteGateway`
- `storagegateway>DeleteSnapshotSchedule`
- `storagegateway>DeleteTape`
- `storagegateway>DeleteTapeArchive`
- `storagegateway>DeleteVolume`

- `storagegateway:DisableGateway`
- `storagegateway:RemoveTagsFromResource`
- `storagegateway:ResetCache`
- `storagegateway:RetrieveTapeArchive`
- `storagegateway:RetrieveTapeRecoveryPoint`
- `storagegateway:SetLocalConsolePassword`
- `storagegateway:ShutdownGateway`
- `storagegateway:StartGateway`
- `storagegateway:UpdateBandwidthRateLimit`
- `storagegateway:UpdateChapCredentials`
- `storagegateway:UpdateGatewayInformation`
- `storagegateway:UpdateGatewaySoftwareNow`
- `storagegateway:UpdateMaintenanceStartTime`
- `storagegateway:UpdateSnapshotSchedule`
- `storagegateway:UpdateVTLDeviceType`

AWS Trusted Advisor

These are the Write actions for AWS Trusted Advisor.

- `trustedadvisor:ExcludeCheckItems`
- `trustedadvisor:IncludeCheckItems`
- `trustedadvisor:RefreshCheck`
- `trustedadvisor:UpdateNotificationPreferences`

AWS WAF

These are the Write actions for AWS WAF.

- `waf>CreateByteMatchSet`
- `waf>CreateIPSet`
- `waf>CreateRateBasedRule`
- `waf>CreateRule`
- `waf>CreateSizeConstraintSet`
- `waf>CreateSqlInjectionMatchSet`
- `waf>CreateXssMatchSet`
- `waf>DeleteByteMatchSet`
- `waf>DeleteIPSet`
- `waf>DeleteRateBasedRule`
- `waf>DeleteRule`
- `waf>DeleteSizeConstraintSet`
- `waf>DeleteSqlInjectionMatchSet`
- `waf>DeleteXssMatchSet`
- `waf>UpdateByteMatchSet`
- `waf>UpdateIPSet`
- `waf>UpdateRateBasedRule`
- `waf>UpdateRule`

- `waf:UpdateSizeConstraintSet`
- `waf:UpdateSqlInjectionMatchSet`
- `waf:UpdateXssMatchSet`

AWS WAF Regional

These are the Write actions for AWS WAF Regional.

- `waf-regional:AssociateWebACL`
- `waf-regional:CreateByteMatchSet`
- `waf-regional:CreateIPSet`
- `waf-regional:CreateRateBasedRule`
- `waf-regional:CreateRule`
- `waf-regional:CreateSizeConstraintSet`
- `waf-regional:CreateSqlInjectionMatchSet`
- `waf-regional:CreateXssMatchSet`
- `waf-regional:DeleteByteMatchSet`
- `waf-regional:DeleteIPSet`
- `waf-regional:DeleteRateBasedRule`
- `waf-regional:DeleteRule`
- `waf-regional:DeleteSizeConstraintSet`
- `waf-regional:DeleteSqlInjectionMatchSet`
- `waf-regional:DeleteXssMatchSet`
- `waf-regional:DisassociateWebACL`
- `waf-regional:UpdateByteMatchSet`
- `waf-regional:UpdateIPSet`
- `waf-regional:UpdateRateBasedRule`
- `waf-regional:UpdateRule`
- `waf-regional:UpdateSizeConstraintSet`
- `waf-regional:UpdateSqlInjectionMatchSet`
- `waf-regional:UpdateXssMatchSet`

Amazon WorkDocs

These are the Write actions for Amazon WorkDocs.

- `workdocs:AbortDocumentVersionUpload`
- `workdocs:ActivateUser`
- `workdocs:AddResourcePermissions`
- `workdocs:AddUserToGroup`
- `workdocs>CreateFolder`
- `workdocs>CreateInstance`
- `workdocs>CreateNotificationSubscription`
- `workdocs>CreateUser`
- `workdocs:DeactivateUser`
- `workdocs>DeleteDocument`
- `workdocs>DeleteFolder`

- `workdocs:DeleteFolderContents`
- `workdocs:DeleteInstance`
- `workdocs:DeleteNotificationSubscription`
- `workdocs:DeleteUser`
- `workdocs:DeregisterDirectory`
- `workdocs:InitiateDocumentVersionUpload`
- `workdocs:RegisterDirectory`
- `workdocs:RemoveAllResourcePermissions`
- `workdocs:RemoveResourcePermission`
- `workdocs:RemoveUserFromGroup`
- `workdocs:UpdateDocument`
- `workdocs:UpdateDocumentVersion`
- `workdocs:UpdateFolder`
- `workdocs:UpdateInstanceAlias`
- `workdocs:UpdateUser`

Amazon WorkMail

These are the Write actions for Amazon WorkMail.

- `workmail:AddMembersToGroup`
- `workmail:CreateGroup`
- `workmail:CreateMailDomain`
- `workmail:CreateMailUser`
- `workmail:CreateOrganization`
- `workmail:CreateResource`
- `workmail:DeleteMailDomain`
- `workmail:DeleteMobileDevice`
- `workmail:DeleteOrganization`
- `workmail:DisableMailGroups`
- `workmail:DisableMailUsers`
- `workmail:EnableMailDomain`
- `workmail:EnableMailGroups`
- `workmail:EnableMailUsers`
- `workmail:RemoveMembersFromGroup`
- `workmail:ResetUserPassword`
- `workmail:SetAdmin`
- `workmail:SetDefaultMailDomain`
- `workmail:SetMailGroupDetails`
- `workmail:SetMailUserDetails`
- `workmail:SetMobilePolicyDetails`
- `workmail:WipeMobileDevice`

Amazon WorkSpaces

These are the Write actions for Amazon WorkSpaces.

- `workspaces:CreateTags`
- `workspaces:CreateWorkspaces`
- `workspaces:DeleteTags`
- `workspaces:ModifyWorkspaceProperties`
- `workspaces:RebootWorkspaces`
- `workspaces:RebuildWorkspaces`
- `workspaces:StartWorkspaces`
- `workspaces:StopWorkspaces`
- `workspaces:TerminateWorkspaces`

Amazon WorkSpaces Application Manager

These are the Write actions for Amazon WorkSpaces Application Manager.

- `wam:AuthenticatePackager`

AWS XRay

These are the Write actions for AWS XRay.

- `xray:PutTelemetryRecords`
- `xray:PutTraceSegments`

Service Actions Included in the Permissions management Access Level

You can use the actions below to grant or modify permissions for AWS resources and IAM principals. If an AWS service does not appear on this page then that service does not have any actions in the Permissions management category.

Topics

- [Auto Scaling \(p. 711\)](#)
- [AWS Certificate Manager \(p. 711\)](#)
- [AWS CloudFormation \(p. 711\)](#)
- [Amazon CloudSearch \(p. 711\)](#)
- [AWS CodeStar \(p. 711\)](#)
- [Amazon Glacier \(p. 711\)](#)
- [Identity And Access Management \(p. 712\)](#)
- [AWS IoT \(p. 712\)](#)
- [AWS Key Management Service \(p. 712\)](#)
- [AWS Lambda \(p. 713\)](#)
- [AWS OpsWorks \(p. 713\)](#)
- [Amazon RDS \(p. 713\)](#)
- [Amazon Redshift \(p. 713\)](#)
- [Amazon S3 \(p. 713\)](#)
- [Amazon SNS \(p. 713\)](#)
- [Amazon SQS \(p. 714\)](#)

- [AWS Service Catalog \(p. 714\)](#)
- [AWS WAF \(p. 714\)](#)
- [AWS WAF Regional \(p. 714\)](#)

Auto Scaling

These are the Permissions management actions for Auto Scaling.

- `autoscaling:DeletePolicy`
- `autoscaling:ExecutePolicy`
- `autoscaling:PutScalingPolicy`

AWS Certificate Manager

These are the Permissions management actions for AWS Certificate Manager.

- `acm>DeleteCertificate`
- `acm:ResendValidationEmail`

AWS CloudFormation

These are the Permissions management actions for AWS CloudFormation.

- `cloudformation:SetStackPolicy`

Amazon CloudSearch

These are the Permissions management actions for Amazon CloudSearch.

- `cloudsearch:UpdateServiceAccessPolicies`

AWS CodeStar

These are the Permissions management actions for AWS CodeStar.

- `codestar:AssociateTeamMember`
- `codestar>CreateProject`
- `codestar>DeleteProject`
- `codestar:DisassociateTeamMember`
- `codestar:UpdateTeamMember`

Amazon Glacier

These are the Permissions management actions for Amazon Glacier.

- `glacier:AbortVaultLock`
- `glacier:CompleteVaultLock`
- `glacier>DeleteVaultAccessPolicy`
- `glacier:InitiateVaultLock`
- `glacier:SetDataRetrievalPolicy`

- `glacier:SetVaultAccessPolicy`

Identity And Access Management

These are the Permissions management actions for Identity And Access Management.

- `iam:AttachGroupPolicy`
- `iam:AttachRolePolicy`
- `iam:AttachUserPolicy`
- `iam:CreatePolicy`
- `iam:CreatePolicyVersion`
- `iam:DeleteAccountPasswordPolicy`
- `iam:DeleteGroupPolicy`
- `iam:DeletePolicy`
- `iam:DeletePolicyVersion`
- `iam:DeleteRolePolicy`
- `iam:DeleteUserPolicy`
- `iam:DetachGroupPolicy`
- `iam:DetachRolePolicy`
- `iam:DetachUserPolicy`
- `iam:PutGroupPolicy`
- `iam:PutRolePolicy`
- `iam:PutUserPolicy`
- `iam:SetDefaultPolicyVersion`
- `iam:UpdateAssumeRolePolicy`

AWS IoT

These are the Permissions management actions for AWS IoT.

- `iot:AttachPrincipalPolicy`
- `iot:CreatePolicy`
- `iot:CreatePolicyVersion`
- `iot:DeletePolicy`
- `iot:DeletePolicyVersion`
- `iot:DetachPrincipalPolicy`
- `iot:SetDefaultPolicyVersion`

AWS Key Management Service

These are the Permissions management actions for AWS Key Management Service.

- `kms>CreateGrant`
- `kms>CreateKey`
- `kms:PutKeyPolicy`
- `kms:RetireGrant`
- `kms:RevokeGrant`

AWS Lambda

These are the Permissions management actions for AWS Lambda.

- `lambda:AddPermission`
- `lambda:EnableReplication`
- `lambda:RemovePermission`

AWS OpsWorks

These are the Permissions management actions for AWS OpsWorks.

- `opsworks:SetPermission`
- `opsworks:UpdateUserProfile`

Amazon RDS

These are the Permissions management actions for Amazon RDS.

- `rds:AuthorizeDBSecurityGroupIngress`

Amazon Redshift

These are the Permissions management actions for Amazon Redshift.

- `redshift:AuthorizeCluster`
- `redshift:AuthorizeSnapshotAccess`
- `redshift>CreateClusterUser`
- `redshift:CreateSnapshotCopyGrant`
- `redshift:JoinGroup`
- `redshift:ModifyClusterIamRoles`
- `redshift:RevokeClusterSecurityGroupIngress`
- `redshift:RevokeSnapshotAccess`
- `redshift:RotateEncryptionKey`

Amazon S3

These are the Permissions management actions for Amazon S3.

- `s3>DeleteBucketPolicy`
- `s3:PutBucketAcl`
- `s3:PutBucketPolicy`
- `s3:PutObjectAcl`
- `s3:PutObjectVersionAcl`

Amazon SNS

These are the Permissions management actions for Amazon SNS.

- `sns:AddPermission`

- `sns:RemovePermission`

Amazon SQS

These are the Permissions management actions for Amazon SQS.

- `sqs:AddPermission`
- `sqs:RemovePermission`

AWS Service Catalog

These are the Permissions management actions for AWS Service Catalog.

- `servicecatalog>CreatePortfolioShare`
- `servicecatalog>DeletePortfolioShare`

AWS WAF

These are the Permissions management actions for AWS WAF.

- `waf>CreateWebACL`
- `waf>DeleteWebACL`
- `waf:UpdateWebACL`

AWS WAF Regional

These are the Permissions management actions for AWS WAF Regional.

- `waf-regional>CreateWebACL`
- `waf-regional>DeleteWebACL`
- `waf-regional:UpdateWebACL`

Resources

IAM is a rich product, and you'll find many resources to help you learn more about how IAM can help you secure your AWS account and resources.

Topics

- [Users and Groups \(p. 715\)](#)
- [Credentials \(Passwords, Access Keys, and MFA devices\) \(p. 715\)](#)
- [Permissions and Policies \(p. 715\)](#)
- [Federation and Delegation \(p. 716\)](#)
- [IAM and Other AWS Products \(p. 716\)](#)
- [General Security Practices \(p. 717\)](#)
- [General Resources \(p. 717\)](#)

Users and Groups

Consult these resources for creating, managing, and using users and groups.

- [Creating Your First IAM Admin User and Group \(p. 17\)](#) – A step-by-step procedure that shows how to create an IAM users and assign permissions.
- [Identities \(Users, Groups, and Roles\) \(p. 59\)](#) – An in-depth discussion of how to administer IAM users and groups.
- [Guidelines for When to Use Accounts, Users, and Groups](#) – An AWS Security Blog post that discusses how to organize user access with separate AWS accounts or with IAM users and groups in a single account.

Credentials (Passwords, Access Keys, and MFA devices)

Review the following guides to manage passwords for your AWS account and for IAM users. You'll also find information about *access keys*—the secret key that you use to make programmatic calls to AWS.

- [AWS Security Credentials](#) – Describes the types of credentials you use to access Amazon Web Services, explains how to create and manage them, and includes recommendations for managing access keys securely.
- [Managing Passwords \(p. 74\)](#) and [Managing Access Keys for IAM Users \(p. 85\)](#) – Describes options for managing credentials for IAM users in your account.
- [Using Multi-Factor Authentication \(MFA\) in AWS \(p. 91\)](#) – Describes how to configure your account and IAM users to require both a password and a one-time use code that is generated on a device before sign-in is allowed. (This is sometimes called two-factor authentication.)

Permissions and Policies

Learn the inner workings of IAM policies and find tips on the best ways to confer permissions:

- [IAM Policies \(p. 275\)](#) – Describes how permissions can be attached to users or groups or, for some AWS products, to resources themselves.
- [IAM Policies \(p. 275\)](#) – Introduces the policy language that is used to define permissions.
- [IAM JSON Policy Elements Reference \(p. 426\)](#) – Provides descriptions and examples of each policy language element.
- [Example Policies \(p. 295\)](#) – Shows examples of policies for common tasks in various AWS products.
- [AWS Policy Generator](#) – Create custom policies by choosing products and actions from a list.
- [IAM Policy Simulator](#) – Test whether a policy would allow or deny a specific AWS action. The following video (6:28) provides an overview and shows the policy simulator in action.

[Getting Started with the IAM Policy Simulator](#)

Federation and Delegation

You can grant access to resources in your AWS account for users who are authenticated (signed in) elsewhere. These can be IAM users in another AWS account (known as *delegation*), users who are authenticated with your organization's sign-in process, or users from an Internet identity provider like Login with Amazon, Facebook, Google, or any other OpenID Connect (OIDC) compatible identity provider. In these cases, the users get temporary security credentials to access AWS resources.

- [Tutorial: Delegate Access Across AWS Accounts Using IAM Roles \(p. 26\)](#) – Guides you through granting cross-account access to an IAM user in another AWS account.
- [Common Scenarios for Temporary Credentials \(p. 232\)](#) – Describes ways in which users can be federated into AWS after being authenticated outside of AWS.
- [Web Identity Federation Playground](#) – Lets you experiment with Login with Amazon, Google, or Facebook to authenticate and then make a call to Amazon S3.

IAM and Other AWS Products

Most AWS products are integrated with IAM so that you can use IAM features to help protect access to the resources in those products. The following resources discuss IAM and security for some of the most popular AWS products. For a complete list of products that work with IAM, including links to more information on each, see [AWS Services That Work with IAM \(p. 417\)](#).

Using IAM with Amazon EC2

- [Controlling Access to Amazon EC2 Resources](#) – Describes how to use IAM features to permit users to administer Amazon EC2 instances, volumes, and more.
- [Using Instance Profiles \(p. 218\)](#) – Describes how to use IAM roles to securely provide credentials for applications that run on Amazon EC2 instances and that need access to other AWS products.

Using IAM with Amazon S3

- [Managing Access Permissions to Your Amazon S3 Resources](#) – Discusses the Amazon S3 security model for buckets and objects, which includes IAM policies.
- [Writing IAM Policies: Grant Access to User-Specific Folders in an Amazon S3 Bucket](#) – Discusses how to let users protect their own folders in Amazon S3. (For more posts about Amazon S3 and IAM, choose the **S3** tag below the title of the blog post.)

Using IAM with Amazon RDS

- [Using AWS Identity and Access Management \(IAM\) to Manage Access to Amazon RDS Resources](#) – Describes how to use IAM to control access to database instances, database snapshots, and more.
- [A Primer on RDS Resource-Level Permissions](#) – Describes how to use IAM to control access to specific Amazon RDS instances.

Using IAM with Amazon DynamoDB

- [Using IAM to Control Access to DynamoDB Resources](#) – Describes how to use IAM to permit users to administer DynamoDB tables and indexes.
- The following video (8:55) explains how to provide access control for individual DynamoDB database items or attributes (or both).

[Getting Started with Fine-Grained Access Control for DynamoDB](#)

General Security Practices

Find expert tips and guidance on the best ways to secure your AWS account and resources:

- [AWS Security Best Practices \(PDF\)](#) – Provides an in-depth look at how to manage security across AWS accounts and products, including suggestions for security architecture, use of IAM, encryption and data security, and more.
- [IAM Best Practices \(p. 43\)](#) – Offers recommendations for ways to use IAM to help secure your AWS account and resources.
- [AWS CloudTrail User Guide](#) – Use AWS CloudTrail to track a history of API calls made to AWS and store that information in log files. This helps you determine which users and accounts accessed resources in your account, when the calls were made, what actions were requested, and more.

General Resources

Explore the following resources to learn more about IAM and AWS.

- [Product Information for IAM](#) – General information about the AWS Identity and Access Management product.
- [Discussion Forums for AWS Identity and Access Management](#) – A community forum for customers to discuss technical questions related to IAM.
- [Classes & Workshops](#) – 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, SDKs, IDE toolkits, and command line tools for developing and managing AWS applications.
- [AWS Whitepapers](#) – Links to a comprehensive list of technical AWS whitepapers, covering topics such as architecture, security, and economics and authored by AWS Solutions Architects or other technical experts.
- [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 web page 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.

Calling the API by Making HTTP Query Requests

Topics

- [Endpoints \(p. 719\)](#)
- [HTTPS Required \(p. 720\)](#)
- [Signing IAM API Requests \(p. 720\)](#)

This section contains general information about using the Query API for AWS Identity and Access Management (IAM) and AWS Security Token Service (AWS STS). For details about the API actions and errors, go to the [IAM API Reference](#) or the [AWS Security Token Service API Reference](#).

Note

Instead of making direct calls to the IAM or AWS STS APIs, 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, etc.). The SDKs provide a convenient way to create programmatic access to IAM and AWS. For example, the SDKs take care of tasks such as cryptographically signing requests (see below), managing errors, and retrying requests automatically. For information about the AWS SDKs, including how to download and install them, see the [Tools for Amazon Web Services](#) page.

The Query API for IAM and AWS STS lets you call service actions. Query API requests are HTTPS requests that must contain an `Action` parameter to indicate the action to be performed. IAM and AWS STS support GET and POST requests for all actions. That is, the API does not 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 [IAM API Reference](#) or the [AWS Security Token Service API Reference](#).

Endpoints

IAM and AWS STS each have a single global endpoint:

- (IAM) <https://iam.amazonaws.com>
- (AWS STS) <https://sts.amazonaws.com>

Note

AWS STS also supports sending requests to regional endpoints in addition to the global endpoint. Before you can use AWS STS in a region, you must first activate STS in that region for your AWS account. For more information about activating additional regions for AWS STS, see [Activating and Deactivating AWS STS in an AWS Region \(p. 257\)](#).

For more information about AWS endpoints and regions for all services, see [Regions and Endpoints](#) in the [AWS General Reference](#).

HTTPS Required

Because the Query API returns sensitive information such as security credentials, you must use HTTPS with all API requests.

Signing IAM API Requests

Requests must be signed using an access key ID and a secret access key. We strongly recommend that you do not use your AWS account root user credentials for everyday work with IAM. You can use the credentials for an IAM user or you can use AWS STS to generate temporary security credentials.

To sign your API requests, we recommend using AWS Signature Version 4. For information about using Signature Version 4, go to [Signature Version 4 Signing Process](#) in the *AWS General Reference*.

If you need to use Signature Version 2, information about using Signature Version 2 is available in the [AWS General Reference](#).

For more information, see the following:

- [AWS Security Credentials](#). Provides general information about the types of credentials used for accessing AWS.
- [IAM Best Practices \(p. 43\)](#). Presents a list of suggestions for using IAM service to help secure your AWS resources.
- [Temporary Security Credentials \(p. 231\)](#). Describes how to create and use temporary security credentials.

AWS Glossary

For the latest AWS terminology, see the [AWS Glossary](#) in the *AWS General Reference*.