**IAM Policies - overview of JSON 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](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create.html) and [Editing IAM Policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_manage-edit.html).
* **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](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html).

## 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](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-services-that-work-with-iam.html). To learn more about resource-based policies, see [Identity-Based Policies and Resource-Based Policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_identity-vs-resource.html).

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](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_permissions-to-switch.html).

**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](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create.html) and [Editing IAM Policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_manage-edit.html).

**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](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements.html).

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](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_identity-vs-resource.html).

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](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_examples.html).

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](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_manage.html). Also, you can use the [AWS Policy Generator](https://awspolicygen.s3.amazonaws.com/policygen.html) 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](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_evaluation-logic.html).

**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.


        General policy structure
      

The information in a statement is contained within a series of elements. For information about these elements, see [IAM JSON Policy Elements Reference](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements.html).

**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"}}

}

]

}