**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. To learn more about the general structure and purpose of a JSON policy document, see [Overview of JSON Policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html#access_policies-json).

Some JSON policy elements are mutually exclusive. This means that you cannot create a policy that uses both. For example, you cannot use both Action and NotAction in the same policy statement. Other pairs that are mutually exclusive include Principal/NotPrincipal and Resource/NotResource.

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](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-services-that-work-with-iam.html).

**Topics**

* [IAM JSON Policy Elements: Version](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_version.html)
* [IAM JSON Policy Elements: Id](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_id.html)
* [IAM JSON Policy Elements: Statement](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_statement.html)
* [IAM JSON Policy Elements: Sid](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_sid.html)
* [IAM JSON Policy Elements: Effect](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_effect.html)
* [IAM JSON Policy Elements: Principal](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_principal.html)
* [IAM JSON Policy Elements: NotPrincipal](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_notprincipal.html)
* [IAM JSON Policy Elements: Action](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_action.html)
* [IAM JSON Policy Elements: NotAction](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_notaction.html)
* [IAM JSON Policy Elements: Resource](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_resource.html)
* [IAM JSON Policy Elements: NotResource](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_notresource.html)
* [IAM JSON Policy Elements: Condition](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_condition.html)
* [Variables and Tags](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_variables.html)
* [Supported Data Types](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_datatypes.html)

# 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[Versioning IAM Policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-versioning.html).

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.

# 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 Statementelement. (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](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_variables.html" \l "policy-vars-intro).

{

"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](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_evaluation-logic.html).

# 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)](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-arns) 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*](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_terms-and-concepts.html#iam-term-service-role). 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](http://docs.aws.amazon.com/general/latest/gr/acct-identifiers.html#FindingCanonicalId) 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](http://docs.aws.amazon.com/AmazonS3/latest/dev/example-bucket-policies.html) in the Amazon Simple Storage Service Developer Guide
* [Example Policies for Amazon SNS](http://docs.aws.amazon.com/sns/latest/dg/UsingIAMwithSNS.html#ExamplePolicies_SNS) in the Amazon Simple Notification Service Developer Guide
* [Amazon SQS Policy Examples](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/SQSExamples.html) in the Amazon Simple Queue Service Developer Guide
* [Key Policies](http://docs.aws.amazon.com/kms/latest/developerguide/key-policies.html) in the AWS Key Management Service Developer Guide
* [Account Identifiers](http://docs.aws.amazon.com/general/latest/gr/acct-identifiers.html) in the AWS General Reference
* [About Web Identity Federation](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_oidc.html)

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

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](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_principal.html) 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](http://docs.aws.amazon.com/AmazonS3/latest/dev/using-with-s3-actions.html) in the Amazon Simple Storage Service Developer Guide, the list of actions for Amazon EC2 can be found in the [Amazon EC2 API Reference](http://docs.aws.amazon.com/AWSEC2/latest/APIReference/query-apis.html), and the list of actions for AWS Identity and Access Management can be found in the [IAM API Reference](http://docs.aws.amazon.com/IAM/latest/APIReference/API_Operations.html). To find the list of actions for other services, consult the API reference [documentation](http://aws.amazon.com/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](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/acp-overview.html#PermissionTypes) 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 NotActionelement 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](http://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html).)

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

# 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 aResource 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](http://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html).)

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 NotResourceelement 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](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_condition_operators.html) (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](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_condition_operators.html)
* For a description of how to handle condition keys that have multiple values, see [Creating a Condition That Tests Multiple Key Values (Set Operations)](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_multi-value-conditions.html)
* For a list of all of the globally available condition keys, see [Available Global Condition Keys](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_condition-keys.html#AvailableKeys).
* For conditions keys that are defined by each service, see [AWS Service Actions and Condition Context Keys for Use in IAM Policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_actionsconditions.html).

## The Condition Block

The following example shows the basic format of a Condition element:

"Condition": {

"DateGreaterThan" : {

"aws:CurrentTime" : "2013-12-15T12:00:00Z"

}

}

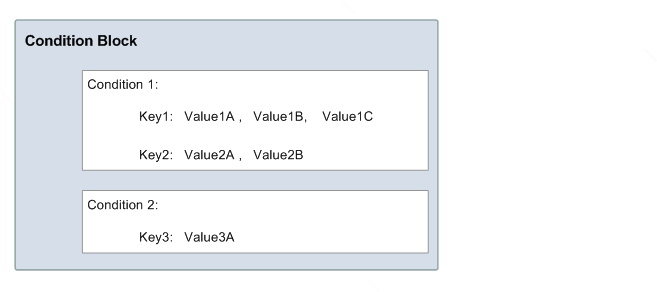
A value from the request is represented by a key, in this case aws:CurrentTime. The key value is compared to a value that you specify either as a literal value (2013-08-16T12:00:00Z) or as a policy variable, as explained later. The type of comparison to make is specified by the [condition operator](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_condition_operators.html) (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 contains 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)](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_multi-value-conditions.html).

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.


          Condition block that includes two NumericEquals conditions
        

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.


          Condition block that includes DateGreaterThan condition
        

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.


          Condition block showing how AND and OR are applied to multiple values
        

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:SourceIpare 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)](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_multi-value-conditions.html).

# IAM Policy Elements: Variables

**Topics**

* [Introduction](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_variables.html#policy-vars-intro)
* [Where You Can Use Policy Variables](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_variables.html#policy-vars-wheretouse)
* [Request Information That You Can Use for Policy Variables](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_variables.html#policy-vars-infotouse)
* [For More Information](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_variables.html#policy-vars-formoreinfo)

## 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](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-friendly-names) 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 [IAM JSON Policy Elements: Version](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_version.html). To learn more about policy versions, see [Versioning IAM Policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-versioning.html).

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 Conditionelement.

### Resource Element

A policy variable can appear as the last part of the [ARN](http://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html) 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](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-unique-ids).

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": "sqs:\*",

"Resource": "arn:aws:sqs: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](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_evaluation-logic.html#policy-eval-reqcontext).

### 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](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp.html).
* **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](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_condition_operators.html#Conditions_IPAddress) 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](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-friendly-names) 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 table shows values for these keys for different types of principal.

|  |  |  |  |
| --- | --- | --- | --- |
| **Principal** | **aws:username** | **aws:userid** | **aws:principaltype** |
| AWS account | (not present) | AWS account ID | Account |
| IAM user | *IAM-user-name* | [unique ID](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-unique-ids) | User |
| Federated user | (not present) | *account*:*caller-specified-name* | FederatedUser |
| Web federated user (Login with Amazon, Amazon Cognito, Facebook, Google)  For information about policy keys that are available when you use web identity federation, see [Identifying Users with Web Identity Federation](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_oidc_user-id.html). | (not present) | *role id*:*caller-specified-role-name*  where role id is the [unique id of the role](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-unique-ids) and the caller-specified-role-name is specified by the [RoleSessionName parameter](http://docs.aws.amazon.com/IAM/latest/APIReference/API_AssumeRole.html" \l "API_AssumeRoleWithWebIdentity_RequestParameters" \t "_blank) passed to the AssumeRoleWithWebIdentity request. | AssumedRole |
| SAML federated user  For information about policy keys that are available when you use SAML federation, see [Uniquely Identifying Users in SAML-Based Federation](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_saml.html#CreatingSAML-userid). | (not present) | *role id*:*caller-specified-role-name*  where role id is the [unique id of the role](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-unique-ids) and the caller-specified-role-name is specified by the Attribute element with the [Name attribute](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_create_saml_assertions.html) set to https://aws.amazon.com/SAML/attributes/RoleSessionName. | AssumedRole |
| Assumed role | (not present) | *role id*:*caller-specified-role-name*  where role id is the [unique id of the role](http://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-unique-ids) and the caller-specified-role-name is specified by the [RoleSessionName parameter](http://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRole.html" \l "API_AssumeRole_RequestParameters" \t "_blank) passed to the AssumeRole request. | AssumedRole |
| Role assigned to an Amazon EC2 instance | (not present) | *role-id*:*ec2-instance-id*  where role id is [the unique id of the role](http://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-unique-ids) and the ec2-instance-id is the [unique identifier of the EC2 instance](http://docs.aws.amazon.com/AWSEC2/latest/APIReference/API_DescribeInstances.html). | AssumedRole |
| Anonymous caller (Amazon SQS, Amazon SNS, and Amazon S3 only) | (not present) | (not present) | Anonymous |

In this table:

* 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](http://docs.aws.amazon.com/mobile/sdkforandroid/developerguide/cognito-auth.html#d0e840)in the AWS Mobile SDK for Android Developer Guide
* [Amazon Cognito Overview](http://docs.aws.amazon.com/mobile/sdkforios/developerguide/cognito-auth.html#d0e664)in the AWS Mobile SDK for iOS Developer Guide
* [Common Scenarios for Temporary Credentials](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp.html#sts-introduction).

### 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](http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingResOpsConditions.html#BucketKeysinAmazonS3Policies) in the Amazon Simple Storage Service Developer Guide.
* [Amazon SNS Keys](http://docs.aws.amazon.com/sns/latest/dg/AccessPolicyLanguage_SpecialInfo.html#sns_aspen_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](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html)
* [Example Policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_examples.html)
* [IAM JSON Policy Elements Reference](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements.html)
* [IAM JSON Policy Evaluation Logic](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_evaluation-logic.html)
* [About Web Identity Federation](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_oidc.html)

**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](http://tools.ietf.org/html/rfc4627).

|  |  |
| --- | --- |
| **Type** | **JSON** |
| String | String |
| Integer | Number |
| Float | Number |
| Boolean | true false |
| Null | null |
| Date | String adhering to the [W3C Profile of ISO 8601](http://www.w3.org/TR/NOTE-datetime) |
| IpAddress | String adhering to [RFC 4632](http://tools.ietf.org/html/rfc4632) |
| List | Array |
| Object | Object |