# Troubleshoot IAM

Use the information here to help you diagnose and fix common issues when you work with AWS Identity and Access Management (IAM).

**Issues**



## I can't sign in to my AWS account

Verify that you have the correct credentials and that you are using the correct method to sign in. For more information, see Troubleshooting sign-in issues in the *AWS Sign-In User Guide*.

## 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, you must delete an existing pair before you can create another.

If you lose your secret access key, you must delete the access key and create a new one. For more instructions, see Update access keys.

## Policy variables aren't working

If your policy variables are not working, one of the following errors has occurred:

**The date is wrong in the Version policy element.**

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. Policies that don't include variables still work when 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 is created when you modify 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. To learn more about policy versions, see the section called “Versioning IAM policies”.

**Variable characters are in the wrong letter case.**

Verify that your policy variables are in the right case. For details, see IAM policy elements: Variables and tags.

## 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 changes that you make in IAM (or other AWS services), including attribute-based access control (ABAC) tags, take time to become visible from all possible endpoints. Some delay results from the time it takes to send data from server to server, replication zone to replication zone, and Region to Region. 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. 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**: Read consistency in the *DynamoDB Developer Guide*, 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 EMR 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 User Guide*

## I am not authorized to perform: iam:DeleteVirtualMFADevice

You might receive the following error when you attempt to assign or remove a virtual MFA device for yourself or others:

```
User: arn:aws:iam::123456789012:user/Diego is not authorized to
 perform: iam:DeleteVirtualMFADevice on resource: arn:aws:iam::123456789012:mfa/Diego
 with an explicit deny
```

This could happen if someone previously began assigning a virtual MFA device to a user in the IAM console and then cancelled the process. This creates a virtual MFA device for the user in IAM but never assigns it to the user. Delete the existing virtual MFA device before you create a new virtual MFA device with the same device name.

To fix this issue, an administrator should **not** edit policy permissions. Instead, the administrator must use the AWS CLI or AWS API to delete the existing but unassigned virtual MFA device.

**To delete an existing but unassigned virtual MFA device**

1. View the virtual MFA devices in your account.
   - AWS CLI: `aws iam list-virtual-mfa-devices`
   - AWS API: `ListVirtualMFADevices`
2. In the response, locate the ARN of the virtual MFA device for the user you are trying to fix.
3. Delete the virtual MFA device.
   - AWS CLI: `aws iam delete-virtual-mfa-device`
   - AWS API: `DeleteVirtualMFADevice`

## How do I securely create IAM users?

If you have employees that require access to AWS, you might choose to create IAM users or use IAM Identity Center for authentication. If you use IAM, AWS recommends that you create an IAM user and securely communicate the credentials to the employee. If you are not physically located next to your employee, use a secure workflow to communicate credentials to employees.

Use the following secure workflow to create a new user in IAM:

1. Create a new user using the AWS Management Console. Choose to grant AWS Management Console access with a generated password. If necessary, select the **Users must create a new password at next sign-in** check box. Do not add a permissions policy to the user until after they have changed their password.
2. After the user is added, copy the sign-in URL, user name, and password for the new user. To view the password, choose **Show**.
3. Send the password to your employee using a secure communications method in your company, such as email, chat, or a ticketing system. Separately, provide your users with the IAM user console link and their user name. Tell the employee to confirm that they can sign in successfully before you will grant them permissions.
4. After the employee confirms, add the permissions that they need. As a security best practice, add a policy that requires the user to authenticate using MFA to manage their credentials.

For an example policy, see AWS: Allows MFA-authenticated IAM users to manage their own credentials on the Security credentials page.

## Additional resources

The following resources can help you troubleshoot as you work with AWS.

- **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. For more information, see Logging IAM and AWS STS API calls with AWS CloudTrail.
- **AWS Knowledge Center** – Find FAQs and links to other resources to help you troubleshoot issues.
- **AWS Support Center** – Get technical support.
- **AWS Premium Support Center** – Get premium technical support.

## Troubleshoot access denied error messages

The following information can help you identify, diagnose, and resolve access denied errors with AWS Identity and Access Management. Access denied errors appear when AWS explicitly or implicitly denies an authorization request.

- An *explicit denial* occurs when a policy contains a `Deny` statement for the specific AWS action.
- An *implicit denial* occurs when there is no applicable `Deny` statement and also no applicable `Allow` statement. Because an IAM policy denies an IAM principal by default, the policy must explicitly allow the principal to perform an action. Otherwise, the policy implicitly denies access. For more information, see The difference between explicit and implicit denies.

When you make a request to a service or resource, multiple policies may apply to the request. Review all applicable policies in addition to the policy specified in the error message.

- If multiple policies of the same policy type deny a request, the access denied error message doesn't specify the number of policies evaluated.
- If multiple policy types deny an authorization request, AWS includes only one of those policy types in the error message.

> **Important**
>
> **Having trouble signing in to AWS?** Make sure that you're on the correct AWS sign-in page for your type of user. If you are the AWS account root user (account owner), you can sign in to AWS using the credentials that you set up when you created the AWS account. If you are an IAM user, your account administrator can give you AWS sign-in credentials. If you need to request support, do not use the feedback link on this page. The form is received by the AWS Documentation team, not Support. Instead, on the Contact Us page choose **Still unable to log into your AWS account** and then choose one of the available support options.

## I get "access denied" when I make a request to an AWS service

- Check if the error message includes the type and Amazon Resource Name (ARN) of the policy responsible for denying access. If this is the case, then check for deny statements for the action in the specified policy. If the policy type is provided but there is no policy ARN, then focus on troubleshooting issues for that policy type: Check for deny statements for the action in policies of the specified type. If the error message doesn't mention the policy type responsible for denying access, use the rest of the guidelines in this section to troubleshoot further.
- Verify that you have the identity-based policy 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 Manage IAM policies.
- If the AWS Management Console returns a message stating that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator provided you with your sign-in credentials or sign-in link.

  The following example error occurs when the `mateojackson` IAM user attempts to use the console to view details about a fictional *`my-example-widget`* resource but does not have the fictional `widgets:`*`GetWidget`* permissions.

  ```
  User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
   widgets:GetWidget on resource: my-example-widget
  ```

  In this case, Mateo must ask his administrator to update his policies to allow access to the *`my-example-widget`* resource using the `widgets:`*`GetWidget`* action.

- Are you trying to access a service that supports resource-based policies, such as Amazon S3, Amazon SNS, or Amazon SQS? If so, verify that the policy specifies you as a principal and grants you access. If you make a request to a service within your account, either your identity-based policies or the resource-based policies can grant you permission. If you make a request to a service in a different account, then both your identity-based policies and the resource-based policies must grant you permission. To view the services that support resource-based policies, see AWS services that work with IAM.
- If your policy includes a condition with a key–value pair, review it carefully. Examples include the `aws:RequestTag/tag-key` global condition key, the AWS KMS `kms:EncryptionContext:encryption_context_key`, and the `ResourceTag/tag-key` condition key supported by multiple services. Make sure that the key name does not match multiple results. Because condition key names are not case sensitive, a condition that checks for a key named `foo` matches `foo`, `Foo`, or `F00`. If your request includes multiple key–value pairs with key names that differ only by case, then your access might be unexpectedly denied. For more information, see IAM JSON policy elements: Condition.
- If you have a permissions boundary, verify that the policy that is used for the permissions boundary allows your request. If your identity-based policies allow the request, but your permissions boundary does not, then the request is denied. A permissions boundary controls the maximum permissions that an IAM principal (user or role) can have. Resource-based policies are not limited by permissions boundaries. Permissions boundaries are not common. For more information about how AWS evaluates policies, see Policy evaluation logic.
- If you are signing requests manually (without using the AWS SDKs), verify that you have correctly signed the request.
- If you're using an Amazon VPC endpoint policy and you get an access denied error that is not logged in AWS CloudTrail, it might be because the VPC endpoint owner account is different from the calling account or target role account.

## I get "access denied" when I make a request with temporary security credentials

- First, make sure that you are not denied access for a reason that is unrelated to your temporary credentials. For more information, see I get "access denied" when I make a request to an AWS service.
- Verify that the service accepts temporary security credentials, see AWS services that work with IAM.

- Verify that your requests are being signed correctly and that the request is well-formed. For details, see your toolkit documentation or Use temporary credentials with AWS resources.
- Verify that your temporary security credentials haven't expired. For more information, see Temporary security credentials in IAM.
- Verify that the IAM user or role has the correct permissions. Permissions for temporary security credentials are derived from an IAM user or role. As a result, the permissions are limited to those that are granted to the role whose temporary credentials you have assumed. For more information about how permissions for temporary security credentials are determined, see Permissions for temporary security credentials.
- If you assumed a role, your role session might be limited by session policies. When you request temporary security credentials programmatically using AWS STS, you can optionally pass inline or managed session policies. Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary credential session for a role. You can pass a single JSON inline session policy document using the `Policy` parameter. You can use the `PolicyArns` parameter to specify up to 10 managed session policies. The resulting session's permissions are the intersection of the role's identity-based policies and the session policies. Alternatively, if your administrator or a custom program provides you with temporary credentials, they might have included a session policy to limit your access.
- If you are an AWS STS federated user principal, your session might be limited by session policies. You create a federated user session by signing in to AWS as an IAM user and then requesting a federation token. For more information, see Requesting credentials through a custom identity broker. If you or your identity broker passed session policies while requesting a federation token, then your session is limited by those policies. The resulting session's permissions are the intersection of your IAM user identity-based policies and the session policies. For more information about session policies, see Session policies.
- 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 `amzn-s3-demo-bucket`.

  JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Sid": "S3BucketPolicy",
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::111122223333:role/MyRole"]},
```

```
    "Action": ["s3:PutObject"],
    "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket/*"]
  }]
}
```

## Access denied error message examples

Most access denied error messages appear in the format `User` *`user`* `is not authorized to perform` *`action`* `on` *`resource`* `because` *`context`*. In this example, *`user`* is the ARN of the principal that is denied access, *`action`* is the service action that the policy denies, and *`resource`* is the ARN of the resource on which the policy acts. The *`context`* field provides additional context about the policy type that denied access. In some cases, it also contains the ARN of the policy which denied access.

When a policy explicitly denies access because the policy contains a `Deny` statement, then AWS includes the phrase `with an explicit deny in a` *`type`* `policy` in the access denied error message. This phrase may also specify the ARN of the policy, as follows: `with an explicit deny in a` *`type`* `policy:` *`policy ARN`*.

When the policy implicitly denies access, then AWS includes the phrase `because no` *`type`* `policy allows the` *`action`* `action` in the access denied error message.

> **Note**
>
> Some AWS services do not support this access denied error message format. The content of access denied error messages can vary depending on the service making the authorization request.

The following examples show the format for different types of access denied error messages.

### Access denied due to a service control policy – implicit denial

1. Check for a missing `Allow` statement for the action in your service control policies (SCPs). For the following example, the action is `codecommit:ListRepositories`.
2. Update your SCP by adding the `Allow` statement. For more information, see Updating an SCP in the *AWS Organizations User Guide*.

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codecommit:ListRepositories
because no service control policy allows the codecommit:ListRespositories action
```

## Access denied due to a service control policy – explicit denial

1. If a policy ARN is provided in the error message, check for a Deny statement for the action in the specified service control policy (SCP). In the example below, the action is `codecommit:ListRepositories`.
2. If no policy ARN is provided in the error message, check for a Deny statement for the action in your SCPs.
3. Update your SCP by removing the Deny statement. For more information, see Update a service control policy (SCP) in the *AWS Organizations User Guide*.

Error message with a policy ARN:

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codecommit:ListRepositories
with an explicit deny in a service control policy:
 arn:aws:organizations::777788889999:policy/o-exampleorgid/service_control_policy/p-
examplepolicyid123
```

Error message without a policy ARN:

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codecommit:ListRepositories
with an explicit deny in a service control policy
```

## Access denied due to a resource control policy – explicit denial

1. If a policy ARN is provided in the error message, check for a Deny statement for the action in the specified resource control policy (RCP). In the example below, the action is `secretsmanager:GetSecretValue`.
2. If no policy ARN is provided in the error message, check for a Deny statement for the action in your RCPs.
3. Update your RCP by removing the Deny statement. For more information, see Update a resource control policy (RCP) in the *AWS Organizations User Guide*.

Error message with a policy ARN:

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 secretsmanager:GetSecretValue
on resource: arn:aws:secretsmanager:us-east-1:123456789012:secret:*
with an explicit deny in a resource control policy:
 arn:aws:organizations::777788889999:policy/o-exampleorgid/resource_control_policy/p-
examplepolicyid456
```

Error message without a policy ARN:

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 secretsmanager:GetSecretValue
on resource: arn:aws:secretsmanager:us-east-1:123456789012:secret:*
with an explicit deny in a resource control policy
```

## Access denied due to a VPC endpoint policy – implicit denial

1. Check for a missing `Allow` statement for the action in your Virtual Private Cloud (VPC) endpoint policies. For the following example, the action is `codecommit:ListRepositories`.
2. Update your VPC endpoint policy by adding the `Allow` statement. For more information, see Update a VPC endpoint policy in the *AWS PrivateLink Guide*.

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codecommit:ListRepositories
because no VPC endpoint policy allows the codecommit:ListRepositories action
```

## Access denied due to a VPC endpoint policy – explicit denial

1. Check for an explicit `Deny` statement for the action in your Virtual Private Cloud (VPC) endpoint policies. For the following example, the action is `codedeploy:ListDeployments`.
2. Update your VPC endpoint policy by removing the `Deny` statement. For more information, see Update a VPC endpoint policy in the *AWS PrivateLink Guide*.

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codedeploy:ListDeployments
on resource: arn:aws:codedeploy:us-east-1:123456789012:deploymentgroup:*
```

```
with an explicit deny in a VPC endpoint policy
```

## Access denied due to a permissions boundary – implicit denial

1. Check for a missing `Allow` statement for the action in your permissions boundary. For the following example, the action is `codedeploy:ListDeployments`.
2. Update your permissions boundary by adding the `Allow` statement to your IAM policy. For more information, see Permissions boundaries for IAM entities and Edit IAM policies.

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codedeploy:ListDeployments
on resource: arn:aws:codedeploy:us-east-1:123456789012:deploymentgroup:*
because no permissions boundary allows the codedeploy:ListDeployments action
```

## Access denied due to a permissions boundary – explicit denial

1. If a policy ARN is provided in the error message, check for a `Deny` statement for the action in the specified permissions boundary. In the example below, the action is `sagemaker:ListModels`.
2. If no policy ARN is provided in the error message, check for a `Deny` statement for the action in the permissions boundary attached to the principal.
3. Update your permissions boundary by removing the `Deny` statement from your IAM policy. For more information, see Permissions boundaries for IAM entities and Edit IAM policies.

Error message with a policy ARN:

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 sagemaker:ListModels
with an explicit deny in a permissions boundary: arn:aws:iam::123456789012:policy/
DeveloperPermissionBoundary
```

Error message without a policy ARN:

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 sagemaker:ListModels
with an explicit deny in a permissions boundary
```

## Access denied due to session policies – implicit denial

1. Check for a missing `Allow` statement for the action in your session policies. For the following example, the action is `codecommit:ListRepositories`.
2. Update your session policy by adding the `Allow` statement. For more information, see Session policies and Edit IAM policies.

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codecommit:ListRepositories
because no session policy allows the codecommit:ListRepositories action
```

## Access denied due to session policies – explicit denial

1. If a policy ARN is provided in the error message, check for a `Deny` statement for the action in the specified session policy. In the example below, the action is `codedeploy:ListDeployments`.
2. If no policy ARN is provided in the error message, check for a `Deny` statement for the action in your session policies.
3. Update your session policy by removing the `Deny` statement. For more information, see Session policies and Edit IAM policies.

Error message with a policy ARN:

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codedeploy:ListDeployments
on resource: arn:aws:codedeploy:us-east-1:123456789012:deploymentgroup:*
with an explicit deny in a session policy: arn:aws:iam::123456789012:policy/
DeveloperSessionPolicy
```

Error message without a policy ARN:

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codedeploy:ListDeployments
on resource: arn:aws:codedeploy:us-east-1:123456789012:deploymentgroup:*
with an explicit deny in a session policy
```

## Access denied due to resource-based policies – implicit denial

1. Check for a missing `Allow` statement for the action in your resource-based policy. For the following example, the action is `secretsmanager:GetSecretValue`.
2. Update your policy by adding the `Allow` statement. For more information, see Resource-based policies and Edit IAM policies.

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 secretsmanager:GetSecretValue
because no resource-based policy allows the secretsmanager:GetSecretValue action
```

## Access denied due to resource-based policies – explicit denial

1. Check for an explicit `Deny` statement for the action in your resource-based policy. For the following example, the action is `secretsmanager:GetSecretValue`.
2. Update your policy by removing the `Deny` statement. For more information, see Resource-based policies and Edit IAM policies.

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 secretsmanager:GetSecretValue
on resource: arn:aws:secretsmanager:us-east-1:123456789012:secret:*
with an explicit deny in a resource-based policy
```

## Access denied due to role trust policies – implicit denial

1. Check for a missing `Allow` statement for the action in your role trust policy. For the following example, the action is `sts:AssumeRole`.
2. Update your policy by adding the `Allow` statement. For more information, see Resource-based policies and Edit IAM policies.

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform: sts:AssumeRole
because no role trust policy allows the sts:AssumeRole action
```

### Access denied due to role trust policies – explicit denial

1. Check for an explicit Deny statement for the action in your role trust policy. For the following example, the action is `sts:AssumeRole`.
2. Update your policy by removing the Deny statement. For more information, see Resource-based policies and Edit IAM policies.

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform: sts:AssumeRole
with an explicit deny in the role trust policy
```

### Access denied due to identity-based policies – implicit denial

1. Check for a missing `Allow` statement for the action in identity-based policies attached to the identity. For the following example, the action is `codecommit:ListRepositories` attached to the role HR.
2. Update your policy by adding the `Allow` statement. For more information, see Identity-based policies and Edit IAM policies.

```
User: arn:aws:iam::123456789012:role/HR is not authorized to perform:
 codecommit:ListRepositories
because no identity-based policy allows the codecommit:ListRepositories action
```

### Access is denied due to identity-based policies – explicit denial

1. If a policy ARN is provided in the error message, check for a Deny statement for the action in the specified policy. In the example below, the action is `codedeploy:ListDeployments`.
2. If no policy ARN is provided in the error message, check for a Deny statement for the action in identity-based policies attached to the identity.
3. Update your policy by removing the Deny statement. For more information, see Identity-based policies and Edit IAM policies.

Error message with a policy ARN:

```
User: arn:aws:iam::123456789012:role/HR is not authorized to perform:
 codedeploy:ListDeployments
```

```
on resource: arn:aws:codedeploy:us-east-1:123456789012:deploymentgroup:*
with an explicit deny in an identity-based policy: arn:aws:iam::123456789012:policy/
HRAccessPolicy
```

Error message without a policy ARN:

```
User: arn:aws:iam::123456789012:role/HR is not authorized to perform:
 codedeploy:ListDeployments
on resource: arn:aws:codedeploy:us-east-1:123456789012:deploymentgroup:*
with an explicit deny in an identity-based policy
```

# Troubleshoot issues with the root user

Use the information here to help you troubleshoot issues related to the root user of an AWS account.

> **Note**
>
> AWS accounts managed using AWS Organizations may have centralized root access enabled for member accounts. These member accounts do not have root user credentials, can't sign in as a root user, and are prevented from recovering the root user password. Contact your administrator if you need to perform a task that requires root user credentials.

## I can't perform tasks that I expect to be able to do when signed in as the account root user

Your account might be a member of an organization in AWS Organizations. Your organizational administrator may have a service control policy (SCP) to limit the permissions of your account. SCPs impact all users, including the root user. For more information, see Service control policies in the *AWS Organizations User Guide*.

## I forgot the root user password for my AWS account

If you're a root user and you have lost or forgot the password for your AWS account, you can reset your password. You must know the email address used to create the AWS account, and you must have access to the email account. For more information, see Reset a lost or forgotten root user password.

### I don't have access to the email for my AWS account

When you create an AWS account, you provide an email address and password. These are the credentials for the AWS account root user. If you aren't sure of the email address associated with your AWS account, search for messages sent from `@signin.aws` or `@verify.signin.aws` to any email address for your organization that might have been used to open the AWS account.

If you know the email address but no longer have access to the email, try to recover access to the email. Use one of the following options to regain access to your email:

- If you own the domain for the email address, you can restore a deleted email address. Alternatively, you can set up a catch-all for your email account. A catch-all collects all messages sent to email addresses that no longer exist in the mail server and redirects them to another email address.
- 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.

If you're still not able to sign in to your AWS account, you can find alternate support options at Contact us.

## Troubleshoot 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 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 both **Visual** and **JSON** editor options. 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 .

**Troubleshooting IAM Policy Topics**



## Troubleshoot using the visual editor

When you create or edit a customer managed policy, you can use information in the **Visual** editor 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”.

## Policy restructuring

When you create a policy, AWS validates, processes, and transforms the policy before storing it. When the policy is retrieved, AWS transforms it back to a human-readable format without changing permissions. 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 Amazon Resource Name (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** editor. If you never change the policy in the **Visual** editor and choose **Next** from the **JSON** editor, the policy is less likely to be restructured. When you use the **Visual** editor, IAM might restructure the policy to optimize its appearance. 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:

| Using this editor option | If you edit your policy | And then choose *Next* from this tab | When you choose *Save changes* |
|---|---|---|---|
| **Visual** | Edited | **Visual** | The policy is restructured |
| **Visual** | Edited | **JSON** | The policy is restructured |
| **Visual** | Not Edited | **Visual** | The policy is restructured |
| **JSON** | Edited | **Visual** | The policy is restructured |

| Using this editor option | If you edit your policy | And then choose *Next* from this tab | When you choose *Save changes* |
|---|---|---|---|
| **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, 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** – You might see different fields to build your ARN based on the resource type. 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. Provide the **Region**, **Account**, and **InstanceId** values for your resource. The policy grants permissions to any instance in your account if you provide your account ID and choose **Any** for the Region and instance ID.
- **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.

## 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 that you allow permissions to only those actions and resources that a user needs. You should create a deny statement 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 Policy evaluation logic.

> **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.

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 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** editor option.

## Reducing the size of your policy in the visual editor

When you use the visual editor to create a policy, IAM creates a JSON document to store your policy. You can view this document by switching to the **JSON** editor option. If this JSON document exceeds the size limit of a policy, the visual editor displays an error message. You will not be able to review and save the policy. To view the IAM limitation on the size of a managed policy, see IAM and STS character limits.

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. 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

You might see a warning in the visual editor 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.

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 must manually type the actions and resource ARNs to complete your policy. You can ignore any warnings and continue. Alternatively, you can choose the **JSON** editor option 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 must manually type the actions and resource ARNs to complete your policy. You can ignore any warnings and continue. Alternatively, you can choose the **JSON** editor option 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 must manually type the actions and resource ARNs to complete your policy. You can ignore any warnings and continue. Alternatively, you can choose the **JSON** editor option 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.
- **Action does not support the visual editor** – If your policy includes a supported service with an unsupported action, you must manually type the actions and resource ARNs to complete your policy. You can ignore any warnings and continue. Alternatively, you can choose the **JSON** editor option 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.

- **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. We recommend that you use the visual editor by selecting from the list of services and actions. Then, complete the resource section according to the prompts. 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 the following issues:

  - Misspelled service, action, and resource type names, such as `s2` instead of `s3` or `ListMyBuckets` instead of `ListAllMyBuckets`
  - Unnecessary text in ARNs, such as `arn:aws:s3: : :*`
  - Missing colons in actions, such as `iam.CreateUser`

  You can evaluate a policy that might include typos by choosing **Next** to review the policy summary. Then, confirm whether the policy provides the permissions you intended.

## Troubleshoot with 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, the service summary, and the action summary. 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 for any policies that are attached to an entity on the **Policy details** page for that policy. You can view the policy summary for managed policies on the

**Policies** page. If AWS is unable to render a summary for a policy, you will see the JSON policy document and 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:
  - `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**
  >
  > Be careful when using these policy elements such as `NotPrincipal`, `NotAction`, and `NotResource`. For information about using policy elements, see IAM JSON policy element reference.

  If you provide mismatched services and resources, you can create a policy that does not provide effective permissions. This can occur when 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. Test your policies with the policy simulator to better understand the policy.

## Policy summary includes unrecognized services, actions, or resource types

In the IAM console, if a policy summary 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).

> **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.

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.
- **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).
- **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).
- **Typo** – AWS checks that the JSON is syntactically correct, and that the policy does not include typos or other errors as part of policy validation.

> **Note**
>
> As a best practice, we recommend that you use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions. We recommend that you open your existing policies and review and resolve any policy validation recommendations.

## Service does not support IAM policy summaries

It is possible for the IAM policy summaries or the visual editor to not support a generally available (GA) service or action. 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 **Feedback for IAM** 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 **Feedback for IAM** 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 element reference.

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 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 Actions, Resources, and Condition Keys for AWS Services.

Check whether your policy defines an action, resource, or condition that does not grant permissions. View the policy summary 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**
- **A resource is defined without an applicable action**
- **A condition is defined without an applicable action**

To view examples of policy summaries that include warnings, see the section called "Policy summary (list of services)".

**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` 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.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:Describe*",
            "Resource": "arn:aws:ec2:us-east-2:111122223333: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.

JSON

```
{
    "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.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Action": "cloudfront:*",
        "Resource": [
            "arn:aws:cloudfront:*",
            "arn:aws:s3:::amzn-s3-demo-bucket"
        ]
        }
    ]
```

```
}
```

This policy provides permissions for all CloudFront actions. But because the policy defines the S3 `amzn-s3-demo-bucket` 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.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudfront:*",
                "s3:CreateBucket",
                "s3:ListBucket*",
                "s3:PutBucket*",
                "s3:GetBucket*"
            ],
            "Resource": [
                "arn:aws:cloudfront:*",
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

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 Actions, Resources, and Condition Keys for AWS Services and choose the service to view service documentation for condition keys.

JSON

```
{
    "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:ListBucketVersions` action and the `s3:ListBucket` 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.

JSON

```
{
    "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.

JSON

```
{
    "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 for that service. In the IAM console, when you view the **Policy details** page, 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.

### Changing policies for your IAM identities based on their activity

You can update policies for your IAM identities (users, groups, and roles) based on their activity. To do this, view your account's events in CloudTrail **Event history**. CloudTrail event logs include detailed event information that you can use to change the policy's permissions.

**A user or role is attempting to perform an action in AWS and that request is denied.**

Consider whether the user or role should have permission to perform the action. If so, you can add the action and even the ARN of the resource that they attempted to access to their policy.

**A user or role has permissions that they are not using.**

Consider removing those permissions from their policy. Make sure that your policies grant the least privilege that is needed to perform only the necessary actions.

For more information about using CloudTrail, see Viewing CloudTrail Events in the CloudTrail Console in the *AWS CloudTrail User Guide*.

## Troubleshoot JSON policy documents

You can diagnose and resolve issues relating to JSON policy documents.

### Validate your policies

When you create or edit a JSON policy, IAM can perform policy validation to help you create an effective policy. IAM identifies JSON syntax errors, while IAM Access Analyzer provides additional policy checks with recommendations to help you further refine your policies. To learn more about policy validation, see IAM policy validation. To learn more about IAM Access Analyzer policy checks and actionable recommendations, see IAM Access Analyzer policy validation.

## I don't have permissions for policy validation in the JSON editor

In the AWS Management Console, you might receive the following error if you do not have permissions to view IAM Access Analyzer policy validation results:

`You need permissions. You do not have the permissions required to perform this operation. Ask your administrator to add permissions.`

To fix this error, ask your administrator to add the `access-analyzer:ValidatePolicy` permission for you.

## More than one JSON policy object

An IAM policy must consist of only one JSON object. You denote an object by placing { } braces around it. You can nest other objects within a JSON object by embedding additional { } braces within the outer pair. A policy must 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*):

JSON

```
{
      "Version":"2012-10-17",
      "Statement":
      {
         "Effect":"Allow",
         "Action":"ec2:Describe*",
         "Resource":"*"
      }
   }
   {
     "Statement": {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-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**):

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": "ec2:Describe*",
          "Resource":"*"
        },
        {
          "Effect": "Allow",
          "Action": "s3:*",
          "Resource": "arn:aws:s3:::amzn-s3-demo-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*):

JSON

```
{
      "Version":"2012-10-17",
      "Statement": {
        "Effect": "Allow",
```

```
        "Action": "ec2:Describe*",
        "Resource": "*"
      },
      "Statement": {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-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**):

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": "ec2:Describe*",
          "Resource":"*"
        },
        {
          "Effect": "Allow",
          "Action": "s3:*",
          "Resource": "arn:aws:s3:::amzn-s3-demo-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 is 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 `Statement`:

JSON

```
{
      "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 are a single list.
- Multiple `Resource` elements: all `Resource` elements are combined internally and treated as if they are a single list.

The policy engine does not allow you to save any policy with syntax errors. Correct the errors in the policy before saving. Review and correct any policy validation recommendations for your policies.

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:

JSON

```
{
      "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*):

JSON

```
{
      "Version":"2012-10-17",
      "Statement": {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
        "Resource": "arn:aws:s3:::amzn-s3-demo-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**):

JSON

```
{
      "Version":"2012-10-17",
      "Statement": {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": [
          "arn:aws:s3:::amzn-s3-demo-bucket",
          "arn:aws:s3:::amzn-s3-demo-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. In comparison, a policy version is created when you change 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. To learn more about policy versions, see the section called “Versioning IAM policies”.

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. But it fails to specify 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**):

JSON

```
{
  "Version":"2012-10-17",
  "Statement":
  {
    "Action": "iam:*AccessKey*",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::123456789012:user/${aws:username}"
  }
}
```

# Troubleshoot Passkeys and FIDO Security Keys

Use the information here to help you diagnose common issues that you might encounter when working with FIDO2 security keys.

**Topics**



## I can't enable my FIDO security key

Consult the following solutions depending on your status as an IAM user or system administrator

### IAM users

If you can't enable your FIDO security key, check the following:

- Are you using a supported configuration?

IAM supports FIDO2 security devices that connect to your devices through USB, Bluetooth, or NFC. IAM also supports platform authenticators such as TouchID or FaceID. IAM does not support local passkey registration for Windows Hello. To create and use passkeys, Windows users should use cross-device authentication where you use a passkey from one device like a mobile device or hardware security key to sign in on another device like a laptop.

For information on devices and browsers you can use with WebAuthn and AWS, see Supported configurations for using passkeys and security keys.

- Are you using any browser plugins?

  AWS does not support the use of plugins to add WebAuthn browser support. Instead, use a browser that offers native support of the WebAuthn standard.

  Even if you're using a supported browser, you may have a plugin that is incompatible with WebAuthn. An incompatible plugin may prevent you from enabling and using your FIDO-compliant security key. Disable any plugins that might be incompatible and restart your browser. Then, retry enabling the FIDO security key.
- Do you have the appropriate permissions?

  If you don't have any of the above compatibility issues, you may not have the appropriate permissions. Contact your system administrator.

### System administrators

If your IAM users can't enable their FIDO security keys despite using a supported configuration, check their permissions. For a detailed example, see IAM tutorial: Permit users to manage their credentials and MFA settings.

## I can't sign in using my FIDO security key

If you can't sign in to the AWS Management Console using your FIDO security key, first see Supported configurations for using passkeys and security keys. If you're using a supported configuration but cannot sign in, contact your system administrator for assistance.

## I lost or broke my FIDO security key

Up to **eight** MFA devices of any combination of the currently supported MFA types can be assigned to a user. With multiple MFA devices, you only need one MFA device to sign in to the AWS

Management Console. Replacing a FIDO security key is similar to replacing a hardware TOTP token. If you lose or break any type of MFA device, see Recover an MFA protected identity in IAM.

### Other issues

If you have an issue with FIDO security keys that is not covered here, do one of the following:

- IAM users: Contact your system administrator.
- AWS account root users: Contact AWS Support.

## Troubleshoot 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

Check the following:

- To allow users to assume the current role again within a role session, specify the role ARN or AWS account ARN as a principal in the role trust policy. AWS services that provide compute resources such as Amazon EC2, Amazon ECS, Amazon EKS, and Lambda provide temporary credentials and automatically update these credentials. This ensures that you always have a valid set of credentials. For these services, it's not necessary to assume the current role again to obtain temporary credentials. However, if you intend to pass session tags or a session policy, you need

to assume the current role again. To learn how to modify a role trust policy to add the principal role ARN or AWS account ARN, see Update a role trust policy .

- When you assume a role using the AWS Management Console, make sure to use the exact name of your role. Role names are case sensitive when you assume a role.
- When you assume a role using AWS STS API or AWS CLI, make sure to use the exact name of your role in the ARN. Role names are case sensitive when you assume a role.
- When you assume a role using a SAML-based federated identity provider and SAML encryption is enabled, make sure you have uploaded a valid private decryption key for the SAML identity provider. For more information, see Manage SAML encryption keys.
- 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. In addition, 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 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 your IAM identity is tagged with any tags that the IAM policy requires. For example, in the following policy permissions, the `Condition` element requires that you, as the principal requesting to assume the role, must have a specific tag. You must be tagged with `department = HR` or `department = CS`. Otherwise, you cannot assume the role. To learn about tagging IAM users and roles, see the section called “Tags for IAM resources”.

```
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "*",
    "Condition": {"StringEquals": {"aws:PrincipalTag/department": [
            "HR",
            "CS"
        ]}}
```

- 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. Consider 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 from which you are calling `AssumeRole` 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 that is attached to the role that you want to assume. In this example, the account ID with the IAM user that 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. It does not 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": {
      "DateGreaterThan": {"aws:CurrentTime": "2017-07-01T00:00:00Z"},
      "DateLessThan": {"aws:CurrentTime": "2017-12-31T23:59:59Z"}
    }
```

- **Source Identity** – Administrators can configure roles to require identities to pass a custom string that identifies the person or application that is performing actions in AWS, called *source identity*. Verify whether the role being assumed requires that a source identity is set. For more information about source identity, see Monitor and control actions taken with assumed roles.

## 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 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 Create a service-linked role.

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 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 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 and look for the services that have **Yes** in the **Service-Linked Role** column.

## I'm not authorized to perform: iam:PassRole

When you create a service-linked role, you must have permission to pass that role to the service. Some services automatically create a service-linked role in your account when you perform an action in that service. For example, Amazon EC2 Auto Scaling creates the `AWSServiceRoleForAutoScaling` service-linked role for you the first time that you create an Auto Scaling group. If you try to create an Auto Scaling group without the `PassRole` permission, you receive the following error:

```
ClientError: An error occurred (AccessDenied) when calling the
PutLifecycleHook operation: User: arn:aws:sts::111122223333:assumed-role/
Testrole/Diego is not authorized to perform: iam:PassRole on resource:
arn:aws:iam::111122223333:role/aws-service-role/autoscaling.amazonaws.com/
AWSServiceRoleForAutoScaling
```

To fix this error, ask your administrator to add the `iam:PassRole` permission for you.

To learn which services support service-linked roles, see AWS services that work with IAM. To learn whether a service automatically creates a service-linked role for you, choose the **Yes** link to view the service-linked role documentation for the service.

## Why can't I assume a role with a 12-hour session? (AWS CLI, AWS API)

When you use the AWS STS `AssumeRole*` API or `assume-role*` CLI operations to assume a role, you can specify a value for the `DurationSeconds` parameter. You can specify a value from 900 seconds (15 minutes) up to the **Maximum session duration** setting for the role. If you specify a value higher than this setting, the operation fails. This setting can have a maximum value of 12 hours. For example, if you specify a session duration of 12 hours, but your administrator set the maximum session duration to 6 hours, your operation fails. To learn how to view the maximum value for your role, see Update the maximum session duration for a role.

If you use *role chaining* (using a role to assume a second role), your session is limited to a maximum of one hour. If you then use the `DurationSeconds` parameter to provide a value greater than one hour, the operation fails.

## I receive an error when I try to switch roles in the IAM console

The information you enter on the **Switch Role** page must match the information for the role. Otherwise, the operation fails and you receive the following error:

```
Invalid information in one or more fields. Check your information or
contact your administrator.
```

If you receive this error, confirm that the following information is correct:

- **Account ID or alias** – The AWS account ID is a 12-digit number. Your account might have an alias, which is a friendly identifier such as your company name that can be used instead of your AWS account ID. You can use either the account ID or the alias in this field.

- **Role name** – Role names are case sensitive. The account ID and role name must match what is configured for the role.

If you continue to receive an error message, contact your administrator to verify the previous information. The role trust policy or the IAM user policy might limit your access. Your administrator can verify the permissions for these policies.

## My role has a policy that allows me to perform an action, but I get "access denied"

Your role session might be limited by session policies. When you request temporary security credentials programmatically using AWS STS, you can optionally pass inline or managed session policies. Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary credential session for a role. You can pass a single JSON inline session policy document using the `Policy` parameter. You can use the `PolicyArns` parameter to specify up to 10 managed session policies. The resulting session's permissions are the intersection of the role's identity-based policies and the session policies. Alternatively, if your administrator or a custom program provides you with temporary credentials, they might have included a session policy to limit your access.

## The service did not create the role's default policy version

A service role is a role that a service assumes to perform actions in your account on your behalf. When you set up some AWS service environments, you must define a role for the service to assume. In some cases, the service creates the service role and its policy in IAM for you. Although you can modify or delete the service role and its policy from within IAM, AWS does not recommend this. The role and policy are intended for use only by that service. If you edit the policy and set up another environment, when the service tries to use the same role and policy, the operation can fail.

For example, when you use AWS CodeBuild for the first time, the service creates a role named `codebuild-RWBCore-service-role`. That service role uses the policy named `codebuild-RWBCore-managed-policy`. If you edit the policy, it creates a new version and saves that version as the default version. If you perform a subsequent operation in AWS CodeBuild, the service might try to update the policy. If it does, you receive the following error:

```
codebuild.amazon.com did not create the default version (V2) of the
codebuild-RWBCore-managed-policy policy that is attached to the codebuild-
```

`RWBCore-service-role role. To continue, detach the policy from any other identities and then delete the policy and the role.`

If you receive this error, you must make changes in IAM before you can continue with your service operation. First, set the default policy version to V1 and try the operation again. If V1 was previously deleted, or if choosing V1 doesn't work, then clean up and delete the existing policy and role.

For more information on editing managed policies, see Editing customer managed policies (console). For more information about policy versions, see Versioning IAM policies.

**To delete a service role and its 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 delete.
4. Choose the **Entities attached** tab to view which IAM users, groups, or roles use this policy. If any of these identities use the policy, complete the following tasks:
    a. Create a new managed policy with the necessary permissions. To ensure that the identities have the same permissions before and after your actions, copy the JSON policy document from the existing policy. Then create the new managed policy and paste the JSON document as described in Creating Policies using the JSON editor.
    b. For each affected identity, attach the new policy and then detach the old one. For more information, see Adding and removing IAM identity permissions.
5. In the navigation pane, choose **Roles**.
6. In the list of roles, choose the name of the role that you want to delete.
7. Choose the **Trust relationships** tab to view which entities can assume the role. If any entity other than the service is listed, complete the following tasks:
    a. Create a new role that trusts those entities.
    b. The policy that you created in the previous step. If you skipped that step, create the new managed policy now.
    c. Notify anyone who was assuming the role that they can no longer do so. Provide them with information about how to assume the new role and have the same permissions.
8. Delete the policy.

9. Delete the role.

## There is no use case for a service role in the console

Some services require that you manually create a service role to grant the service permissions to perform actions on your behalf. If the service is not listed in the IAM console, you must manually list the service as the trusted principal. If the documentation for the service or feature that you are using does not include instructions for listing the service as the trusted principal, provide feedback for the page.

To manually create a service role, you must know the service principal for the service that will assume the role. A service principal is an identifier that is used to grant permissions to a service. The service principal is defined by the service.

You can find the service principal for some services by checking the following:

1. Open AWS services that work with IAM.
2. Check whether the service has **Yes** in the **Service-linked roles** column.
3. Choose the **Yes** link to view the service-linked role documentation for that service.
4. Find the Service-linked role permissions section for that service to view the service principal.

You can manually create a service role using AWS CLI commands or AWS API operations. To manually create a service role using the IAM console, complete the following tasks:

1. Create an IAM role using your account ID. Do not attach a policy or grant any permissions. For details, see Create a role to give permissions to an IAM user.
2. Open the role and edit the trust relationship. Instead of trusting the account, the role must trust the service. For example, update the following `Principal` element:

   ```
   "Principal": { "AWS": "arn:aws:iam::123456789012:root" }
   ```

   Change the principal to the value for your service, such as IAM.

   ```
   "Principal": { "Service": "iam.amazonaws.com" }
   ```

3. Add the permissions that the service requires by attaching permissions policies to the role.

4. Return to the service that requires the permissions and use the documented method to notify the service about the new service role.

# Troubleshoot IAM and Amazon EC2

The following information can help you troubleshoot IAM issues with Amazon EC2.

**Topics**



## When I try to launch an instance, I don't see the role 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. For information about adding permissions to a user, see Manage IAM policies.

  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 using the IAM CLI or API, verify the following:
  - You created an instance profile and added the role to that instance profile.
  - You used the same name for the role and the instance profile. If you name your role and instance profile differently, you won't see the correct role name 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 Use instance profiles.

> **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

The role in the instance profile might have been replaced recently. If so, your application will need to wait for the next automatically scheduled credential rotation before credentials for your role become available.

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.

## 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?. For information about adding permissions to a user, see Manage IAM policies.

## Amazon EC2: When I try 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 IAM role creation.

For more information about the permissions necessary to work with roles, see see How do I get started?. For information about adding permissions to a user, see Manage IAM policies.

## I can't access the temporary security credentials on my EC2 instance

To access temporary security credentials on your EC2 instance, you must first use the IAM console to create a role. Then you launch an EC2 instance that uses that role and examine the running instance. For more information, see **How Do I Get Started?** in Use an IAM role to grant permissions to applications running on Amazon EC2 instances.

If you still can't access your temporary security credentials on your 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 the EC2 `DescribeInstances` API operation or using the `aws ec2 describe-instances` CLI command.

```
[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? 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 must attach a valid instance profile to your Amazon EC2 instance.

If an instance profile with that name exists, check that the instance profile wasn't deleted and another was created with the same name:

1. Call the IAM `GetInstanceProfile` operation to get the `InstanceProfileId`.
2. Call the Amazon EC2 `DescribeInstances` operation to get the `IamInstanceProfileId` for the instance.
3. Verify that the `InstanceProfileId` from the IAM operation matches the `IamInstanceProfileId` from the Amazon EC2 operation.

If the IDs are different, then the instance profile attached to your instances is no longer valid. You must attach a valid instance profile to the instance.

### 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.

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.

### 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.

JSON

```
{"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.

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.

## Troubleshoot IAM and Amazon S3

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 User 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 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.

To do that, modify the bucket policy to allow root user access from the Amazon S3 console or the AWS CLI. Use the following principal, replacing *123456789012* with the ID of the AWS account.

```
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

# Troubleshoot SAML federation with IAM

Use the information here to help you diagnose and fix issues that you might encounter when working with SAML 2.0 and federation with AWS Identity and Access Management.

**Topics**

- Error: Specified provider doesn't exist.
- Error: Requested DurationSeconds exceeds MaxSessionDuration set for this role.
- Error: Private key limit of 2 is reached.
- Error: Response does not contain the required audience.

## 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

The error can also occur when the SAML attribute values sent by the identity provider have either a leading or trailing whitespace, or other invalid characters in the SAML attribute values. For more information on the expected values for SAML attributes, see Configure SAML assertions for the authentication response

For more information, see Configure SAML assertions for the authentication response. To view the SAML response in your browser, follow the steps listed in View a SAML response in your browser.

## 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 Configure SAML assertions for the authentication response. To view the SAML response in your browser, follow the steps listed in View a SAML response in your browser.

## 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. Make sure to use the exact name of your role, because role names are case sensitive. Correct the name of the role in the SAML service provider configuration.

You are allowed access only if your role trust policy includes the `sts:AssumeRoleWithSAML` action. If your SAML assertion is configured to use the `PrincipalTag` attribute, your trust policy must also include the `sts:TagSession` action. For more information about session tags, see Pass session tags in AWS STS.

This error can occur if you do not have `sts:SetSourceIdentity` permissions in your role trust policy. If your SAML assertion is configured to use the `SourceIdentity` attribute, then your trust policy must also include the `sts:SetSourceIdentity` action. For more information about source identity, see Monitor and control actions taken with assumed roles.

This error can also occur if a federated principal does 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 Enabling SAML 2.0 federated principals to access the AWS Management Console and Configure SAML assertions for the authentication response. To view the SAML response in your browser, follow the steps listed in View a SAML response in your browser.

## 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 Configure SAML assertions for the authentication response. To view the SAML response in your browser, follow the steps listed in View a SAML response in your browser.

## Error: Source Identity must match [a-zA-Z_0-9+=,.@-]{2,64} and not begin with "aws:" (service: AWSSecurityTokenService; status code: 400; error code: InvalidIdentityToken)

This error can occur if the `sourceIdentity` attribute value is too long or contains invalid characters. The maximum valid length is 64 characters. For more information about source identity, see Monitor and control actions taken with assumed roles.

For more information about creating SAML assertions, see Configure SAML assertions for the authentication response. To view the SAML response in your browser, follow the steps listed in View a SAML response in your browser.

## 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: Invalid private key.

This error can occur if you do not format your private key file properly. This error may provide additional detail about why the private key is invalid:

- Key is encrypted.
- Key format is not recognized. Private key file must be a .pem file.

When you Create a SAML identity provider in IAM in the AWS Management Console, you must download the private key from your identity provider to provide to IAM to enable encryption. The private key must be a .pem file that uses AES-GCM or AES-CBC encryption algorithm to decrypt SAML assertions.

## Error: Failed to remove private key.

This error can occur when SAML encryption is set to Required, and your request would remove the only private decryption key for the IAM SAML provider. For more information about rotating private keys, see Manage SAML encryption keys.

## Error: Failed to remove private key because the Key ID does not match a private key.

This error can occur if the `keyId` value for the private key does not match either Key ID for the identity provider's private key files.

When you use update-saml-provider or UpdateSAMLProvider API operations to remove SAML encryption private keys, the value in `RemovePrivateKey` must be a valid Key ID for a private key attached to your identity provider.

## 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. The metadata file was uploaded to AWS when you created the identity provider in IAM.

## Error: Could not parse metadata.

This error can occur if you do not format your metadata file properly.

When you create or manage a SAML identity provider in the AWS Management Console, you must retrieve the SAML metadata document from your identity provider.

This metadata file includes the issuer name, expiration information, and keys that can be used to validate the SAML authentication response (assertions) received from the IdP. The metadata file must be encoded in UTF-8 format without a byte order mark (BOM). To remove the BOM, you can encode the file as UTF-8 using a text editing tool, such as Notepad++.

The X.509 certificate included as part of the SAML metadata document must use a key size of at least 1024 bits. Also, the X.509 certificate must also be free of any repeated extensions. You can

use extensions, but the extensions can only appear once in the certificate. If the X.509 certificate does not meet either condition, IdP creation fails and returns an "Unable to parse metadata" error.

As defined by the SAML V2.0 Metadata Interoperability Profile Version 1.0, IAM does not evaluate or take action on the expiration of X.509 certificates in SAML metadata documents. If you are concerned about expired X.509 certificates, we recommend monitoring certificate expiration dates and rotating certificates according to your organization's governance and security policies.

## Error: Unable to update identity provider. No updates are defined for metadata or encryption assertion.

This error can occur if you use the `update-saml-provider` CLI or `UpdateSAMLProvider` API operations, but don't provide update values in your request parameters. For more information on updating your IAM SAML provider, see Create a SAML identity provider in IAM.

## Error: Unable to set assertion encryption mode to Required because no private key is provided.

This error can occur when you haven't previously uploaded a private decryption key and you attempt to set SAML encryption to Required without including a private key in your request.

Make sure that a private key is defined for your IAM SAML provider when using `create-saml-provider` CLI, `CreateSAMLProvider` API, `update-saml-provider` CLI, or `UpdateSAMLProvider` API operations to require encrypted SAML assertions.

## Error: Unable to add and remove private keys in the same request. Set a value for only one of the two parameters.

This error can occur if both add and remove private key values are included in the same request.

When you use update-saml-provider or UpdateSAMLProvider API operations to rotate SAML encryption private key files, you can only add or remove a private key in your request. If you add a private key while removing a private key, the operation fails. For more information about rotating private keys, see Manage SAML encryption keys.

## Error: Specified provider doesn't exist.

This error can occur if the name of the provider in the SAML assertion does not match the name of the provider in IAM. For more information about viewing the provider name, see Create a SAML identity provider in IAM.

## Error: Requested DurationSeconds exceeds MaxSessionDuration set for this role.

This error can occur if you assume a role from the AWS CLI or API.

When you use the assume-role-with-saml CLI or AssumeRoleWithSAML API operations to assume a role, you can specify a value for the `DurationSeconds` parameter. You can specify a value from 900 seconds (15 minutes) up to the maximum session duration setting for the role. If you specify a value higher than this setting, the operation fails. For example, if you specify a session duration of 12 hours, but your administrator set the maximum session duration to 6 hours, your operation fails. To learn how to view the maximum value for your role, see Update the maximum session duration for a role.

## Error: Private key limit of 2 is reached.

This error can occur if you try to add a private key to your identity provider.

You can save up to two private keys for each identity provider. When you use update-saml-provider or UpdateSAMLProvider API operations to add a third private key, the operation fails.

Remove private keys that have expired before adding a new private key. For more information about rotating private keys, see Manage SAML encryption keys.

## Error: Response does not contain the required audience.

This error can occur if there is a mismatch between the audience URL and the identity provider in the SAML configuration. Make sure that your identity provider (IdP) relying party identifier exactly matches the audience URL (entity ID) provided in the SAML configuration.