AWS Account Management

Reference Guide

AWS Account Management: Reference Guide

Table of Contents

Welcome to the AWS Account Management Reference Guide

What is an AWS account?

This guide contains information about AWS accounts. How to create them, how to manage them, and how to use them.

An account in AWS is a fundamental part of accessing AWS services. It serves these two basic functions:

- **Container** – An AWS account is the basic container for all the AWS resources you can create as an AWS customer. When you create an Amazon Simple Storage Service (Amazon S3) bucket or Amazon Relational Database Service (Amazon RDS) database to store your data, or an Amazon Elastic Compute Cloud (Amazon EC2) instance to process your data, you are creating a resource in your account. Every resource is uniquely identified by an Amazon Resource Name (ARN) that includes the account ID of the account that contains, or owns, the resource.
- **Security boundary** – An AWS account is also the basic security boundary for your AWS resources. Resources that you create in your account are available only to users who have credentials for that same account.

  Among the key resources you can create in your account are *identities*, such as IAM users and roles. These identities have credentials that someone can use to sign in, or *authenticate* to AWS. Identities also have permission policies that specify what the person who signed in is authorized to do with the resources in the account.

  You can create an AWS Identity and Access Management (IAM) user to grant access for a person in your company. That IAM user can have a *password* that lets the person access the AWS console. The user can also have an *access key* to let the person run commands from the AWS Command Line Interface (AWS CLI) or invoke APIs from one of the AWS SDKs.

  IAM roles are particularly flexible because you can associate them with external people by using federation and an identity provider, such as AWS IAM Identity Center (successor to AWS Single Sign-On) (IAM Identity Center). If you already have an identity provider in use by your company, you can use it with federation to simplify how you provide access to the resources in your AWS account. AWS supports identity providers that are compatible with industry standards OpenID Connect (OIDC) or SAML 2.0 (Security Assertion Markup Language 2.0). The latter makes any Active Directory implementation a source identity provider if you combine it with Microsoft Active Directory Federation Services.

Do I need multiple AWS accounts?

AWS accounts serve as the fundamental security boundary in AWS. They serve as a resource container that provides a useful level of isolation. The ability to isolate resources and users is a key requirement to establishing a secure, well governed environment.

Separating your resources into separate AWS accounts helps you to support the following principles in your cloud environment:

- **Security control** – Different applications can have different security profiles, requiring different control policies and mechanisms around them. For example, it’s far easier to talk to an auditor and be able to point to a single AWS account that hosts all elements of your workload that are subject to Payment Card Industry (PCI) Security Standards.
- **Isolation** – An AWS account is a unit of security protection. Potential risks and security threats should be contained within an AWS account without affecting others. There could be different security needs due to different teams or different security profiles.
- **Many teams** – Different teams have their different responsibilities and resource needs. You can prevent teams from interfering with each other by moving them to separate AWS accounts.
- **Data isolation** – In addition to isolating the teams, it's important to isolate the data stores to an account. This can help limit the number of people that can access and manage that data store. This helps contain exposure to highly private data and therefore can help in compliance with the European Union's General Data Protection Regulation (GDPR).
- **Business process** – Different business units or products may have completely different purposes and processes. With multiple AWS accounts, you can support a business unit's specific needs.
- **Billing** – An account is the only true way to separate items at a billing level. Multiple accounts help separate items at a billing level across business units, functional teams, or individual users. You can still get all of your bills consolidated to a single payer (using AWS Organizations and consolidated billing) while having line items separated by AWS account.
- **Quota allocation** – AWS service quotas are enforced separately for each AWS account. Separating workloads into different AWS accounts prevents them from consuming quotas for each other.

All of the recommendations and procedures described in this document are in compliance with the AWS Well-Architected Framework. This framework is intended to help you design a flexible, resilient, and scalable cloud infrastructure. Even when you are starting small, we recommend that you proceed in compliance with this guidance in the framework. Doing so can help you scale your environment securely and without impacting your ongoing operations as you grow.

Managing multiple AWS accounts

Before you start adding multiple accounts, you'll want to develop a plan to manage them. For that, we recommend that you use AWS Organizations, which is a free AWS service to manage all of the AWS accounts in your organization.

AWS also offers AWS Control Tower, which adds layers of AWS managed automation to Organizations and automatically integrates it with other AWS services like AWS CloudTrail, AWS Config, Amazon CloudWatch, AWS Service Catalog, and others. These services can incur additional costs. For more information, see AWS Control Tower pricing.

Getting started: Are you a first-time AWS user?

For a step-by-step walk through of creating an AWS account, and then creating your first admin user by using AWS Identity and Access Management (IAM), see the section called “Getting started” (p. 3).

If you're a first-time user of AWS, then your first step is to sign up for an AWS account. When you do this, AWS creates the new AWS account with the details that you provide and assigns it to you.

A brand new AWS account begins with only its built-in *root user*, the intrinsic administrator for the account. You can sign in to the AWS Management Console as the root user by using the email address and password that you provided when you signed-up.

Important

We strongly recommend that you use the root user for ***only*** the following tasks:

- Creating your first administrative user in AWS Identity and Access Management (IAM). You can then use this IAM admin user to perform your administrative tasks instead of the root user. The Getting started tutorial (p. 3) shows you how to create this first user.
- Performing tasks that *only* the root user can perform. For a list of these tasks, see Tasks that require root user credentials (p. 40).
- Protecting your root user credentials by using the recommended best practices (p. 46).

For all other tasks, sign in to the AWS Management Console or the AWS Command Line Interface (AWS CLI) using the credentials of one of the following:

- An IAM user with attached permission policies that allow the required tasks.
- An IAM role with attached permission policies that allow the required tasks. You access the role using some form of federation, such as by using AWS IAM Identity Center (successor to AWS Single Sign-On) (IAM Identity Center), or a SAML 2.0 provider such as Microsoft Active Directory Federation Services.

After you sign in as the root user for the very first time, we recommend that you enable multi-factor authentication (MFA) to help secure this critical user.

Next, you can create an IAM user that has administrator permissions in your AWS account. This user can do almost anything in your account, except for a few tasks that are restricted to only the root user (p. 40).

Getting started with an AWS account

Follow these steps to create your AWS account. After you create your account, sign in as the root user to create an IAM user to use for your daily administrative tasks.

Steps

Prerequisites

To sign up for an AWS account, you need to provide the following information:

- **An account name** – The name of the account appears in several places, such as on your invoice, and in consoles such as the Billing and Cost Management dashboard and the AWS Organizations console.

  We recommend that you use an account naming standard so that the account name can be easily recognized and distinguished from other accounts that you might own. If it's a company account, consider using a naming standard such as *organization-purpose-environment* (for example, *AnyCompany-audit-prod*). If it's a personal account, consider using a naming standard such as *first name-last name-purpose* (for example, *paulo-santos-testaccount*).

  You can change the account name in your account settings after you sign up. For more information, see How do I change the name on my AWS account?
- **An email address** – This email address is used as the sign-in name for the account's root user, and is required for account recovery, such as if you forget the password. You must be able to receive email

messages that are sent to this address. Before you can perform certain tasks, you must verify that you have access to email sent to this address.

> **Important**
> If this account is for a business, we recommend that you use a corporate distribution list (for example, `it.admins@example.com`). Avoid using an individual's corporate email address (for example, `paulo.santos@example.com`). This approach helps to ensure that your company can retain access to the AWS account even when an employee changes positions or leaves the company. The email address can be used to reset the account's root user credentials. Be sure that you protect access to this distribution list or address.

- **A phone number** – This number can be used when confirmation of account ownership is required. You must be able to receive calls at this phone number.

> **Important**
> If this account is for a business, we recommend that you use a corporate phone number rather than a personal phone number. This helps to ensure that your company can retain access to the AWS account even when an employee changes positions or leaves the company.

- **A multi-factor authentication device.** – To secure your AWS resources, enable multi-factor authentication (MFA) on the root account.

Step 1: Create your AWS account

1. Open the AWS home page in your browser.
2. Choose **Create an AWS account**.

   > **Note**
   > If you signed in to AWS recently, choose **Sign in to the Console**. If the option **Create a new AWS account** isn't visible, first choose **Sign in to a different account**, and then choose **Create a new AWS account**.

3. Enter your account information, and then choose **Continue**. Be sure that you enter your account information correctly, especially your email address. If you enter your email address incorrectly, you can't access your account.
4. Choose **Personal** or **Professional**. The difference between these options is only in the information that we ask you for. Both account types have the same features and functions.
5. Enter your company or personal information. Refer to the recommendations in the Prerequisites (p. 3) section about the email address and phone number.
6. Read and accept the AWS Customer Agreement. Be sure that you read and understand the terms of the AWS Customer Agreement.
7. Choose **Create Account and Continue**.

   At this point, you'll receive an email message to confirm that your AWS account is ready to use. You can sign in to your new account by using the email address and password you provided during sign up. However, you can't use any AWS services until you finish activating your account.
8. On the **Payment Information** page, enter the information about your payment method. If you want to use a different address for billing purposes than you provided in Step 3 (p. 4), choose **Use a new address** and then enter the address to use for billing purposes.
9. Choose **Verify and Add**.

   > **Note**
   > If your contact address is in India, your user agreement for your account is with Amazon Internet Services Private Limited (AISPL), a local AWS seller in India. You must provide your CVV as part of the verification process. You might also have to enter a one-time password, depending on your bank. AISPL charges your payment method 2 INR as part of the verification process. AISPL refunds the 2 INR after it completes verification.

10. Next, you must verify your phone number. Choose your country or region code from the list, and enter a phone number where you can be called in the next few minutes. Enter the CAPTCHA code, and submit.
11. The AWS automated verification system calls you and provides a PIN. Enter the PIN using your phone and then choose **Continue**.
12. Finally, you can select your AWS Support plan. Choose one of the available plans. For a description of the available plans, see Compare AWS Support plans.

    A confirmation page appears that indicates that your account is being activated. This usually takes only a few minutes but can sometimes take up to 24 hours. During activation, you can sign in to your new AWS account. Until activation is complete, you might see a **Complete Sign Up** button. You can ignore it.

    AWS sends a confirmation email message when account activation is complete. Check your email and spam folder for the confirmation email message. After you receive this message, you have full access to all AWS services.

Troubleshooting delays in account activation

Account activation can sometimes be delayed. If the process takes more than 24 hours, check the following:

- **Finish the account activation process.**

  You might have accidentally closed the window for the sign-up process before you added all the necessary information. To finish the sign-up process, open the registration page. Then, choose **Sign in to an existing AWS account**, and sign in using the email address and password you chose for the account.
- **Check the information associated with your payment method.** Check Payment Methods in the AWS Billing and Cost Management console. Fix any errors in the information.
- **Contact your financial institution.** Financial institutions occasionally reject authorization requests from AWS. Contact your payment method's issuing institution, and ask that they approve authorization requests from AWS.

  Note
  AWS cancels the authorization request as soon as it's approved by your financial institution. You aren't charged for authorization requests from AWS. Authorization requests might still appear as a small charge (usually 1 USD) on statements from your financial institution.
- **Check your email for requests for additional information.** Check your email and spam folder to see if AWS needs any information from you to complete the activation process.
- **Try a different browser.**
- **Contact AWS Support.** Contact AWS Support for help. Be sure to mention any troubleshooting steps that you already tried

  Note
  Don't provide sensitive information, such as credit card numbers, in any correspondence with AWS.

Step 2: Sign in to your new account's root user

After you successfully create your account, you can sign in and begin to use AWS services.

To sign in to your new account as the root user, see Signing in as the AWS account root user (p. 35).

Step 3: Enable multi-factor authentication for your root user

We strongly recommend that you enable MFA on the root user of your account. This dramatically lowers the risk of someone being able to access your account without your authorization. For more information, see Activate MFA on the AWS account root user (p. 36).

Step 4: Create an IAM administrator user

Because you can't restrict what a root user can do, we strongly recommend that you don't use your root user for any tasks that don't explicitly require the root user. Instead, create an IAM user that has administrative permissions, and sign in as that IAM user for your daily administrative tasks.

For instructions on how to create such a user, see Creating your first IAM admin user in the *IAM User Guide*.

Managing your AWS account

This section includes topics that describe how to manage your AWS account.

Note
If your AWS account was created in India by using Amazon Internet Services Private Limited (AISPL), there are additional considerations. For more information, see Managing accounts in India (p. 32).

Topics

Creating an AWS account

This topic describes how to create a standalone AWS account that isn't managed by AWS Organizations. If you want to create an account that's part of an organization managed by AWS Organizations, see Creating an AWS account in your organization.

These instructions are for creating an AWS account outside of India. For creating an account in India, see Creating an AWS account with AISPL (p. 33).

AWS Management Console

To create an AWS account

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- Because this operation occurs before you have an AWS account, this operation requires no AWS permissions.

1. Open the Amazon Web Services home page .
2. Choose **Create an AWS account**.

   Note
   If you signed in to AWS recently, that option might not be there. Instead, choose **Sign in to the Console**. Then, if **Create a new AWS account** still isn't visible, first choose **Sign in to a different account**, and then choose **Create a new AWS account**.

3. Enter your account information, and then choose **Continue**. Be sure that you enter your account information correctly, especially your email address. If you enter your email address incorrectly, you won't be able to access your AWS account.

   Important
   Because of the critical nature of the AWS account root user of the account, we strongly recommend that you use an email address that can be accessed by a group, rather than only an individual. That way, if the person who signed up for the AWS account leaves the company, the AWS account can still be used because the email address is still accessible.
   If you lose access to the email address associated with the AWS account, then you can't recover access to the account if you ever lose the password.
4. Choose **Personal** or **Professional**.

   Note
   Personal accounts and professional accounts have the same features and functions.
5. Enter your company or personal information.

   Important
   For professional AWS accounts, it's a best practice to enter a company phone number rather than a number for a personal phone. Configuring the account's Using the AWS account root user (p. 35) with an individual email address or a personal phone number can make your account insecure.
6. Read and accept the AWS Customer Agreement.

   Note
   Be sure that you read and understand the terms of the AWS Customer Agreement.
7. Choose **Create Account and Continue**.
8. On the **Payment Information** page, enter the information about your payment method, and then choose **Verify and Add**.

   Note
   If you want to use a different billing address for your AWS billing information, choose **Use a new address** before you choose **Verify and Add**.

   You can't proceed with thee sign-up process until you add a valid payment method.
9. Next, you must verify your phone number. Choose your country or region code from the list, and enter a phone number where you can be reached in the next few minutes.
10. Enter the code displayed in the CAPTCHA, and then submit.
11. When the automated system contacts you, enter the PIN you receive and then choose **Continue**.
12. On the **Select a Support Plan** page, choose one of the available AWS Support plans. For a description of the available Support plans and their benefits, see Compare AWS Support plans.
13. Finally, wait for your new account to be activated. This usually takes a few minutes but can take up to 24 hours.

    When your account is fully activated, you receive a confirmation email message. Check your email and spam folder for the confirmation message. After you receive this email message, you have full access to all AWS services.

AWS CLI & SDKs

You can create member accounts in an organization that is managed by AWS Organizations by running the CreateAccount operation while signed in to the organization's management account.

You can't create a standalone AWS account outside of an organization by using an AWS Command Line Interface (AWS CLI) or AWS API operation.

Viewing account identifiers

AWS assigns the following unique identifiers to each AWS account:

AWS account ID (p. 9)

A 12-digit number, such as 123456789012, that uniquely identifies an AWS account. Many AWS resources include the account ID in their Amazon Resource Names (ARNs). The account ID portion distinguishes resources in one account from the resources in another account. If you're an AWS Identity and Access Management (IAM) user, you can sign in to the AWS Management Console using either the account ID or account alias.

Canonical user ID (p. 10)

An alpha-numeric identifier, such as `79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be`, that is an obfuscated form of the AWS account ID. You can use this ID to identify an AWS account when granting cross-account access to buckets and objects using Amazon Simple Storage Service (Amazon S3). You can retrieve the canonical user ID for your AWS account as either the root user or an IAM user.

You must be authenticated with AWS to view these identifiers.

Warning
Do not provide your AWS credentials (including passwords and access keys) to a third party that needs your AWS account identifiers to share AWS resources with you. Doing so would give them the same access to the AWS account that you have.

Finding your AWS account ID

You can find the AWS account ID using either the AWS Management Console or the AWS Command Line Interface (AWS CLI). In the console, the location of the account ID depends on whether you're signed in as the root user or an IAM user. The account ID is the same whether you're signed in as the root user or an IAM user.

Finding your account ID as the root user

AWS Management Console

To find your AWS account ID when signed in as the root user

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- When you sign in as the root user, you don't need any IAM permissions.

1. In the navigation bar on the upper right, choose your account name or number and then choose **My Security Credentials**.
2. Expand the **Account identifiers** section. The account number appears next to the label **AWS account ID**.

AWS CLI & SDKs

To find your AWS account ID using the AWS CLI

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- When you run the command as the root user, you don't need any IAM permissions.

Use the get-caller-identity command as follows.

```
$ aws sts get-caller-identity \
    --query Account \
    --output text
123456789012
```

Finding your account ID as an IAM user

AWS Management Console

To find your AWS account ID when signed in as an IAM user

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- `aws-portal:ViewAccount`

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

   Tip
   If you don't see the **My Security Credentials** page, you might be signed in as a federated user with an IAM role, instead of as an IAM user. In this case, look for the entry **My Account** and the account ID number next to it.
2. At the top of the page, under **Account details**, the account number appears next to the label **AWS account ID**.

AWS CLI & SDKs

To find your AWS account ID using the AWS CLI

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- When you run the command as an IAM user or role, then you must have:
  - `sts:GetCallerIdentity`

Use the get-caller-identity command as follows.

```
$ aws sts get-caller-identity \
    --query Account \
    --output text
123456789012
```

Finding the canonical user ID for your AWS account

You can find the canonical user ID for your AWS account using the AWS Management Console or the AWS CLI. The canonical user ID for an AWS account is specific to that account. You can retrieve the canonical user ID for your AWS account as the root user, a federated user, or an IAM user.

Finding the canonical ID as the root user or IAM user

AWS Management Console

To find the canonical user ID for your account when signed in to the console as the root user or an IAM user

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- When you run the command as the root user, you don't need any IAM permissions.
- When you sign in as an IAM user, then you must have:
  - `aws-portal:ViewAccount`

1. Sign in to the AWS Management Console as the root user or an IAM user.
2. In the navigation bar on the upper right, choose your account name or number and then choose **My Security Credentials**.
3. If you are the root user, expand **Account identifiers** and find **Canonical User ID**.

   You will see the AWS Account ID and the Canonical User ID values listed. You can use your canonical user ID to configure Amazon S3 access control lists (ACLs).

   If you're an IAM user, under **Account details**, find **Account canonical user ID**.

AWS CLI & SDKs

To find the canonical user ID using the AWS CLI

The same AWS CLI and API command works for the AWS account root user, IAM users, or IAM roles.

Use the list-buckets command as follows.

```
$ aws s3api list-buckets \
    --query Owner.ID \
    --output text
249fa2f1dc32c330EXAMPLE91b2778fcc65f980f9172f9cb9a5f50ccbEXAMPLE
```

Finding the canonical ID as a federated user with an IAM role

AWS Management Console

To find the canonical user ID for your account when signed in to the console as a federated user with an IAM role

Minimum permissions

- You must have permission to list and view an Amazon S3 bucket.

1. Sign in to the AWS Management Console as a federated user with an IAM role.
2. In the Amazon S3 console, choose a bucket name to view details about a bucket.
3. Choose **Permissions**, and then choose **Access Control List**.

   At the top of the page, under **Access for bucket owner**, the canonical user ID for the AWS account appears.

AWS CLI & SDKs

To find the canonical user ID using the AWS CLI

The same AWS CLI and API command works for the AWS account root user, IAM users, or IAM roles.

Use the list-buckets command as follows.

```
$ aws s3api list-buckets \
    --query Owner.ID \
    --output text
249fa2f1dc32c330EXAMPLE91b2778fcc65f980f9172f9cb9a5f50ccbEXAMPLE
```

Modifying the account name, email address, or password for the AWS account root user

To edit your AWS account's name, or to change the root user's password or email address, perform the steps in the following procedure. The email address and password are the credentials you use to sign in as the AWS account root user.

Note
Changes to an AWS account can take up to four hours propagate everywhere.

AWS Management Console

To edit your AWS account name, root user password, or root user email address

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- You must sign in as the AWS account root user, which requires no additional IAM permissions. You can't perform these steps as an IAM user or role.

1. Use your AWS account's email address and password to sign in to the AWS Management Console as your AWS account root user.
2. Choose your account name on the top right of the window, and then choose **My Account**.
3. On the **Account Settings** page, next to **Account Settings**, choose **Edit**.

   Note
   You might be prompted to approve your access to this information. AWS sends a request to the email address associated with the account and to the primary contact phone number. Choose the link in the request to open it in your browser, and approve the access.
4. Next to the field that you want to update, choose **Edit**.
5. After you have entered your changes, choose **Save changes**.
6. After you have made all of your changes, choose **Done**.

AWS CLI & SDKs

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

Understanding API modes of operation

The API operations that work with an AWS account's attributes always work in one of two modes of operation:

- **Standalone context** – this mode is used when a user or role in an account accesses or changes an account attribute in the ***same account***. The standalone context mode is automatically used when you ***don't*** include the `AccountId` parameter when you call one of the Account Management AWS CLI or AWS SDK operations.
- **Organizations context** – this mode is used when a user or role in one account in an organization accesses or changes an account attribute in a different member account in the same organization. The organizations context mode is automatically used when you ***do*** include the `AccountId` parameter when you call one of the Account Management AWS CLI or AWS SDK operation. You can call the operations in this mode from only the management account of the organization, or the delegated admin account for Account Management.

The AWS CLI and AWS SDK operations can work in either standalone or organizations context.

- If you ***don't*** include the `AccountId` parameter, then the operation runs in the standalone context and automatically applies the request to the account you used to make the request. This is true whether or not the account is a member of an organization.
- If you do include the `AccountId` parameter, then the operation runs in the organizations context, and the operation works on the specified Organizations account.
  - If the account calling the operation is the management account or the delegated admin account for the Account Management service, then you can specify any member account of that organization in the `AccountId` parameter to update the specified account.
  - The only account in an organization that can call one of the alternate contact operations and specify its own account number in the `AccountId` parameter is the account specified as the delegated admin account (p. 42) for the Account Management service. Any other account, including the management account, receives an `AccessDenied` exception.
- If you run an operation in standalone mode, then you must be permitted to run the operation with an IAM policy that includes a `Resource` element of either `"*"` to allow all resources, or an ARN that uses the syntax for a standalone account (p. 13).
- If you run an operation in organizations mode, then you must be permitted to run the operation with an IAM policy that includes a `Resource` element of either `"*"` to allow all resources, or an ARN that uses the syntax for a member account in an organization (p. 14).

Granting permissions to update account attributes

As with most AWS operations, you grant permissions to add, update, or delete account attributes for AWS accounts by using IAM permission policies. When you attach an IAM permission policy to an IAM principal (either a user or role), you specify which actions that principal can perform on which resources, and under what conditions.

The following are some Account Management specific considerations for creating a permissions policy.

Amazon Resource Name format for AWS accounts

- The Amazon Resource Name (ARN) for an AWS account that you can include in the `resource` element of a policy statement is constructed differently based on whether the account you want to reference is a standalone account or an account that is in an organization. See the previous section on Understanding API modes of operation (p. 13).
  -

An account ARN for a standalone account:

```
arn:aws:account::{AccountId}:account
```

You must use this format when you run an account attributes operation in standalone mode by not including the `AccountID` parameter.

- An account ARN for a member account in an organization:

```
arn:aws:account::{ManagementAccountId}:account/o-{OrganizationId}/{AccountId}
```

You must use this format when you run an account attributes operation in organizations mode by including the `AccountID` parameter.

Context keys for IAM policies

The Account Management service also provides several Account Management service-specific condition keys (p. 60) that provide fine-grained control over the permissions you grant.

`account:AccountResourceOrgPaths`

The context key `account:AccountResourceOrgPaths` lets you specify a path through your organization's hierarchy to a specific organizational unit (OU). Only member accounts that are contained by that OU match the condition. The following example snippet restricts the policy to apply to only accounts that are in either of two specified OUs.

Because `account:AccountResourceOrgPaths` is a multi-valued string type, you must use the `ForAnyValue` or `ForAllValues` multi-value string operators. Also, note that the prefix on the condition key is `account`, even though you're referencing paths to OUs in an organization.

```
"Condition": {
    "ForAnyValue:StringLike": {
        "account:AccountResourceOrgPaths": [
            "o-aa111bb222/r-a1b2/ou-a1b2-f6g7h111/*",
            "o-aa111bb222/r-a1b2/ou-a1b2-f6g7h222/*"
        ]
    }
}
```

`account:AccountResourceOrgTags`

The context key `account:AccountResourceOrgTags` lets you reference the tags that can be attached to an account in an organization. A tag is a key/value string pair that you can use to categorize and label the resources in your account. For more information about tagging, see Tag Editor in the *AWS Resource Groups User Guide*. For information about using tags as part of an attribute-based access control strategy, see What is ABAC for AWS in the *IAM User Guide*. The following example snippet restricts the policy to apply to only accounts in an organization that have the tag with the key `project` and a value of either `blue` or `red`.

Because `account:AccountResourceOrgTags` is a multi-valued string type, you must use the `ForAnyValue` or `ForAllValues` multi-value string operators. Also, note that the prefix on the condition key is `account`, even though you're referencing the tags on an organization's member account.

```
"Condition": {
    "ForAnyValue:StringLike": {
        "account:AccountResourceOrgTags/project": [
            "blue",
            "red"
        ]
    }
}
```

Note
You can attach tags to only an account in an organization. You can't attach tags to a standalone AWS account.

Updating contact information

You can store contact information about the primary account contact (p. 20) for your AWS account. You can also add or edit contact information for the following alternate account contacts (p. 15):

- **Billing** – The alternate billing contact will receive billing-related notifications, such as invoice availability notifications.
- **Operations** – The alternate operations contact will receive operations-related notifications.
- **Security** – The alternate security contact will receive security-related notifications, including notifications from the AWS Abuse Team.

Topics

Accessing or updating the alternate contacts

You can update alternate contacts for accounts within your organization from the AWS Organizations console, or programmatically with AWS CLI or AWS SDKs. You can use the organization’s management account to view and edit account settings for any account in your organization. The primary account holder will continue to receive all email communications to the root account’s email.

You can add or edit alternate contacts differently, depending on whether or not the accounts are standalone, or part of an organization:

- **Standalone AWS accounts** – For member accounts not associated with an organization, you can update your own alternate contacts using the AWS Management Console, or via AWS CLI & SDKs. To learn how to do this, see Update standalone AWS account alternate contacts (p. 16).
- **AWS accounts within an organization** – The management account user can update any account in the organization from the AWS Organizations console, or programmatically via the AWS CLI & SDKs. To learn how to do this, see Update AWS account alternate contacts in your organization (p. 18).

Context keys for IAM policies

`account:AlternateContactTypes`

The context key `account:AlternateContactTypes` lets you specify which of the three billing types is allowed (or denied) by the IAM policy.

For example, the following example IAM permission policy uses this condition key to allow the attached principals to retrieve, but not modify, only the `BILLING` alternate contact for a specific account in an organization.

Because `account:AlternateContactTypes` is a multi-valued string type, you must use the `ForAnyValue` or `ForAllValues` multi-value string operators.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "account:GetAlternateContact",
            "Resource": [
                "arn:aws:account::123456789012:account/o-aa111bb222/111111111111"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "account:AlternateContactTypes": [
                        "BILLING"
                    ]
                }
            }
        }
    ]
}
```

Update standalone AWS account alternate contacts

To edit your AWS account's alternate contact details, perform the steps in the following procedure.

The AWS Management Console procedure below always works *only* in the standalone context. You can use the AWS Management Console to access or change only the alternate contacts in the account you used to call the operation.

AWS Management Console

To edit your standalone AWS account's alternate contact details

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- `aws-portal:ViewAccount` (to see the account details page)

You must also have one of the following permission options:
The following permission lets a user perform any or all of the alternate contact commands:

- `aws-portal:ModifyAccount`

1. Sign in to the AWS Management Console as an IAM user or role that has the minimum permissions.
2. Choose your account name on the top right of the window, and then choose **My account**.
3. On the **Account Settings** page, scroll down to **Alternate Contacts**, and to the right of the title, choose **Edit**.
4. Change the values in any of the available fields.

   Important
   For professional AWS accounts, it's a best practice to enter a company phone number and email address rather than one belonging to an individual.
5. After you have made all of your changes, choose **Update**.

AWS CLI & SDKs

You can retrieve, update, or delete the ***alternate*** contact information can be by using the following AWS CLI commands or their AWS SDK equivalent operations:

- GetAlternateContact
- PutAlternateContact
- DeleteAlternateContact

Notes

- To perform these operations from the management account or a delegated admin account in an organization against member accounts, you must enable trusted access for the Account service.

Minimum permissions

For each operation, you must have the permission that maps to that operation:

- `account:GetAlternateContact`
- `account:PutAlternateContact`
- `account:DeleteAlternateContact`

If you use these individual permissions, you can grant some users the ability to only read the contact information, and grant others the ability to both read and write.
The `aws-portal` permissions apply to only the AWS Management Console, and can't be used to grant permissions for the AWS CLI or AWS SDK operations.

Example

The following example retrieves the current Billing alternate contact for the caller's account.

```
$ aws account get-alternate-contact \
    --alternate-contact-type=BILLING
{
    "AlternateContact": {
        "AlternateContactType": "BILLING",
        "EmailAddress": "saanvi.sarkar@amazon.com",
        "Name": "Saanvi Sarkar",
        "PhoneNumber": "+1(206)555-0123",
        "Title": "CFO"
    }
}
```

Example

The following example sets a new Operations alternate contact for the caller's account.

```
$ aws account put-alternate-contact \
    --alternate-contact-type=OPERATIONS \
    --email-address=mateo_jackson@amazon.com \
    --name="Mateo Jackson" \
    --phone-number="+1(206)555-1234" \
    --title="Operations Manager"
```

This command produces no output if it's successful.

Example

> **Note**
> If you perform multiple `PutAlternateContact` operations on the same AWS account and the same contact type, the first adds the new contact, and all successive calls to the same AWS account and contact type update the existing contact.

Example

The following example deletes the Security alternate contact for the caller's account.

```
$ aws account delete-alternate-contact \
    --alternate-contact-type=SECURITY
```

This command produces no output if it's successful.

> **Note**
> If you try to delete the same contact more than once, the first succeeds silently. All later attempts generate a `ResourceNotFound` exception.

Update AWS account alternate contacts in your organization

To edit your AWS account's alternate contact details, perform the steps in the following procedure.

Requirements

To update alternate contacts with the AWS Organizations console, you need to do some preliminary settings:

- Your organization must enable all features to manage settings on your member accounts. This allows admin control over the member accounts. This is set by default when you create your organization. If your organization is set to consolidated billing only, and you want to enable all features," see Enabling all features in your organization.
- You need to enable trusted access for AWS Account Management service. To set this up, see Enabling trusted access for AWS Account Management.

> **Note**
> The AWS Organizations managed policies `AWSOrganizationsReadOnlyAccess` or `AWSOrganizationsFullAccess` are updated to provide permission to access the AWS Account Management APIs so you can access account data from the AWS Organizations console. To view the updated managed policies, see Updates to Organizations AWS managed policies.

AWS Management Console

To edit your AWS account's alternate contact details in an organization

1. Sign in to the AWS Organizations console with the organization's management account credentials.
2. From **AWS accounts**, select the account that you want to update.
3. Choose **Contact info**, and under **Alternate contacts**, locate the type of contact: **Billing contact**, **Security contact**, or **Operations contact**.
4. To add a new contact, select **Add**, or to update an existing contact select **Edit**.
5. Change the values in any of the available fields.

   > **Important**
   > For professional AWS accounts, it's a best practice to enter a company phone number and email address rather than one belonging to an individual.

6. After you have made all of your changes, choose **Update**.

AWS CLI & SDKs

You can retrieve, update, or delete the ***alternate*** contact information can be by using the following AWS CLI commands or their AWS SDK equivalent operations:

- GetAlternateContact
- PutAlternateContact
- DeleteAlternateContact

Notes

- To perform these operations from the management account or a delegated admin account in an organization against member accounts, you must enable trusted access for the Account service.
- You can't access an account in a different organization from the one you're using to call the operation.

Minimum permissions
For each operation, you must have the permission that maps to that operation:

- `account:GetAlternateContact`
- `account:PutAlternateContact`
- `account:DeleteAlternateContact`

If you use these individual permissions, you can grant some users the ability to only read the contact information, and grant others the ability to both read and write.
The `aws-portal` permissions apply to only the AWS Management Console, and can't be used to grant permissions for the AWS CLI or AWS SDK operations.

Example

The following example retrieves the current Billing alternate contact for the caller's account in an organization. The credentials used must be from either the organization's management account, or from the Account Management's delegated admin account.

```
$ aws account get-alternate-contact \
    --alternate-contact-type=BILLING \
    --account-id 123456789012
{
    "AlternateContact": {
        "AlternateContactType": "BILLING",
        "EmailAddress": "saanvi.sarkar@amazon.com",
        "Name": "Saanvi Sarkar",
        "PhoneNumber": "+1(206)555-0123",
        "Title": "CFO"
    }
}
```

Example

The following example sets the Operations alternate contact for the specified member account in an organization. The credentials used must be from either the organization's management account, or from the Account Management's delegated admin account.

```
$ aws account put-alternate-contact \
    --account-id 123456789012 \
    --alternate-contact-type=OPERATIONS \
    --email-address=mateo_jackson@amazon.com \
    --name="Mateo Jackson" \
    --phone-number="+1(206)555-1234" \
    --title="Operations Manager"
```

This command produces no output if it's successful.

Note
If you perform multiple `PutAlternateContact` operations on the same AWS account and the same contact type, the first adds the new contact, and all successive calls to the same AWS account and contact type update the existing contact.

Example

The following example deletes the Security alternate contact for the specified member account in an organization. The credentials used must be from either the organization's management account, or from the Account Management's delegated admin account.

```
$ aws account delete-alternate-contact \
    --account-id 123456789012 \
    --alternate-contact-type=SECURITY
```

This command produces no output if it's successful.

Example

Note
If you try to delete the same contact more than once, the first succeeds silently. All later attempts generate a `ResourceNotFound` exception.

Accessing or updating the primary account contact

You can update the primary account contact for accounts within your organization programmatically with AWS CLI or AWS SDKs.

You can add or edit the primary account contact differently, depending on whether or not the accounts are standalone, or part of an organization:

- **Standalone AWS accounts** – For accounts not associated with an organization, you can update your own primary account contact using the AWS Management Console, or via AWS CLI & SDKs. To learn how to do this, see Update standalone AWS account primary contact (p. 20).
- **AWS accounts within an organization** – A user in the management account or delegated admin account can update any account in the organization from the AWS Organizations console, or programmatically via the AWS CLI & SDKs. To learn how to do this, see Update AWS account primary contact in your organization (p. 22).

Update standalone AWS account primary contact

To edit your AWS account's primary contact details, perform the steps in the following procedure.

The AWS Management Console procedure below always works *only* in the standalone context. You can use the AWS Management Console to access or change only the primary contact information of the account you used to call the operation.

AWS Management Console

To edit your AWS account's primary contact details

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- `aws-portal:ViewAccount` (to see the account details page)

- `aws-portal:ModifyAccount`

1. Sign in to the AWS Management Console as an IAM user or role that has the minimum permissions.
2. Choose your account name on the top right of the window, and then choose **My account**.
3. Scroll down to the section **Contact information**, and next to it choose **Edit**.
4. Change the values in any of the available fields.

   Important
   For professional AWS accounts, it's a best practice to enter a company phone number and email address rather than one belonging to an individual. Configuring the account's Using the AWS account root user with an individual's email address or phone number can make your account difficult to recover if that individual leaves the company.
5. After you have made all of your changes, choose **Update**.

AWS CLI & SDKs

You can retrieve, update, or delete the ***primary*** contact information by using the following AWS CLI commands or their AWS SDK equivalent operations:

- GetContactInformation
- PutContactInformation

Notes

- To perform these operations from the management account or a delegated admin account in an organization against member accounts, you must enable trusted access for the Account service.

Minimum permissions
For each operation, you must have the permission that maps to that operation:

- `account:GetContactInformation`
- `account:PutContactInformation`

If you use these individual permissions, you can grant some users the ability to only read the contact information, and grant others the ability to both read and write.
The `aws-portal` permissions apply to only the AWS Management Console, and can't be used to grant permissions for the AWS CLI or AWS SDK operations.

Example

The following example retrieves the current primary contact information for the caller's account.

```
$ aws account get-contact-information
{
    "ContactInformation": {
        "AddressLine1": "123 Any Street",
        "City": "Seattle",
        "CompanyName": "Example Corp, Inc.",
        "CountryCode": "US",
        "DistrictOrCounty": "King",
        "FullName": "Saanvi Sarkar",
        "PhoneNumber": "+15555550100",
        "PostalCode": "98101",
        "StateOrRegion": "WA",
        "WebsiteUrl": "https://www.examplecorp.com"
    }
}
```

Example

The following example sets new primary contact information for the caller's account.

```
$ aws account put-contact-information --contact-information \
'{"AddressLine1": "123 Any Street", "City": "Seattle", "CompanyName": "Example Corp,
 Inc.", "CountryCode": "US", "DistrictOrCounty": "King",
"FullName": "Saanvi Sarkar", "PhoneNumber": "+15555550100", "PostalCode": "98101",
 "StateOrRegion": "WA", "WebsiteUrl": "https://www.examplecorp.com"}'
```

This command produces no output if it's successful.

Update AWS account primary contact in your organization

To edit your AWS account's primary contact details, perform the steps in the following procedure.

Requirements

To update primary contact with the AWS Organizations console, you need to do some preliminary settings:

- Your organization must enable all features to manage settings on your member accounts. This allows admin control over the member accounts. This is set by default when you create your organization. If your organization is set to consolidated billing only, and you want to enable all features," see Enabling all features in your organization.
- You need to enable trusted access for AWS Account Management service. To set this up, see Enabling trusted access for AWS Account Management.

AWS Management Console

You can't currently modify the ***primary*** contact information using the Organizations console. You can modify the primary contact information only by using the AWS Account Console or the AWS CLI and SDKs.

AWS CLI & SDKs

You can retrieve, update, or delete the ***primary*** contact information by using the following AWS CLI commands or their AWS SDK equivalent operations:

- GetContactInformation

- PutContactInformation

Notes

- To perform these operations from the management account or a delegated admin account in an organization against member accounts, you must enable trusted access for the Account service.
- You can't access an account in a different organization from the one you're using to call the operation.

Minimum permissions
For each operation, you must have the permission that maps to that operation:

- `account:GetContactInformation`
- `account:PutContactInformation`

If you use these individual permissions, you can grant some users the ability to only read the contact information, and grant others the ability to both read and write.
The `aws-portal` permissions apply to only the AWS Management Console, and can't be used to grant permissions for the AWS CLI or AWS SDK operations.

Example

The following example retrieves the current primary contact information for the specified member account in an organization. The credentials used must be from either the organization's management account, or from the Account Management's delegated admin account.

```
$ aws account get-contact-information --account-id 123456789012
{
    "ContactInformation": {
        "AddressLine1": "123 Any Street",
        "City": "Seattle",
        "CompanyName": "Example Corp, Inc.",
        "CountryCode": "US",
        "DistrictOrCounty": "King",
        "FullName": "Saanvi Sarkar",
        "PhoneNumber": "+15555550100",
        "PostalCode": "98101",
        "StateOrRegion": "WA",
        "WebsiteUrl": "https://www.examplecorp.com"
    }
}
```

Example

The following example sets the primary contact information for the specified member account in an organization. The credentials used must be from either the organization's management account, or from the Account Management's delegated admin account.

```
$ aws account put-contact-information --account-id 123456789012 \
--contact-information '{"AddressLine1": "123 Any Street", "City": "Seattle",
 "CompanyName": "Example Corp, Inc.", "CountryCode": "US", "DistrictOrCounty": "King",
"FullName": "Saanvi Sarkar", "PhoneNumber": "+15555550100", "PostalCode": "98101",
 "StateOrRegion": "WA", "WebsiteUrl": "https://www.examplecorp.com"}'
```

This command produces no output if it's successful.

Setting or changing security challenge questions

You can improve the security of your account by adding security challenge questions and their answers to your AWS account. AWS can use the information to help authenticate you as the owner of the account when you need to contact AWS Support for help.

To select the security challenge questions and provide the answers, perform the steps in the following procedure.

AWS Management Console

To add or edit security challenge questions for your AWS account

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- `aws-portal:ViewAccount` (to see the account details page)
- `aws-portal:ModifyAccount`

1. Sign in to the AWS Management Console as either the AWS account root user or as an IAM user or role that has the minimum permissions.
2. Choose your account name on the top right of the window, and then choose **My Account**.
3. Scroll down to the section **Configure Security Challenge Questions** and choose **Edit**.
4. Change the values in any of the available fields. You can select any of the provided questions, and then enter the appropriate answer.

   Important
   For professional AWS accounts, it's a best practice to choose the generic **Security Challenge Response #n**.
   For each question, type a value such as a long, random string. Be sure to document and securely store a copy of this information so it can be retrieved by an account administrator when it is needed.
5. After you complete your changes, choose **Update**.

AWS CLI & SDKs

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

Specifying which AWS Regions your account can use

AWS originally enabled all new AWS Regions by default, which enabled your users to create resources in any Region. Now, when AWS adds a Region, the new Region is disabled by default. If you want your users to be able to create resources in a new Region, you enable the Region.

Important
AWS recommends that you use regional AWS Security Token Service (AWS STS) endpoints instead of the global endpoint to reduce latency. Session tokens from regional AWS STS endpoints are valid in all AWS Regions. If you use regional AWS STS endpoints, you don't need to make any changes.
However, session tokens from the *global* AWS STS endpoint (https://sts.amazonaws.com) are valid only in AWS Regions that you enable, or that are enabled by default. If you intend to enable a new Region for your account, you can either use session tokens from regional AWS STS

endpoints or activate the global AWS STS endpoint to issue session tokens that are valid in all AWS Regions. Session tokens that are valid in all Regions are larger. If you store session tokens, these larger tokens might affect your systems.
For more information about how AWS STS endpoints work with AWS Regions, see Managing AWS STS in an AWS Region.

Considerations about enabling and disabling AWS Regions

- **You can use IAM permissions to control access to Regions**

  AWS Identity and Access Management (IAM) includes three permissions that let you control which users can enable, disable, and list Regions. For more information, see Billing and Cost Management actions policies in the *AWS Billing and Cost Management User Guide*.
- **Enabling a Region is free**

  There is no charge to enable a Region. You're charged only for resources that you create in the new Region.
- **Disabling a Region disables access to resources in the Region**

  If you disable a Region that still contains AWS resources, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, you lose access to the resources in that Region. For example, you can't use the AWS Management Console or any programmatic method to view or change the configuration of any EC2 instances in a disabled Region.
- **Charges for active resources continue if you disable a Region**

  If you disable a Region that still contains AWS resources, charges for those resources (if any) continue to accrue at the standard rate. For example, if you disable a Region that contains Amazon EC2 instances, you still have to pay the charges for those instances even though the instances are inaccessible.
- **Disabling a Region isn't always immediately visible**

  If you disable a Region, the change takes time to become visible in all possible endpoints. Disabling a Region can take between a few seconds to minutes to take effect.
- **Original Regions are enabled by default**

  The original AWS Regions (the Regions that existed before we added the ability to enable and disable Regions) are all enabled by default and can't be disabled. For more information, see Managing AWS Regions in the *AWS General Reference*.
- **Enabling a Region takes a few minutes for most accounts**

  Enabling a Region generally takes effect in a few minutes, although it can take longer for some accounts. If enabling a Region takes longer than nine hours, sign in to the AWS Support Center and open a case with AWS Support.

Use the following procedure to enable or disable the AWS Regions for the users in an AWS account.

AWS Management Console

To modify which AWS Regions can be accessed by the AWS account

Minimum permissions

To perform the steps in the following procedure, an IAM user or role must have the following permissions:

- `aws-portal:ViewAccount` (needed to view the account details page)
- `account:ListRegions` (needed to view the list of AWS Regions and whether they are currently enabled or disabled).

- `account:EnableRegion`
- `account:DisableRegion`

1. Sign in to the AWS Management Console as either the AWS account root user or as an IAM user or role that has the minimum permissions.
2. Choose your account name on the top right of the window, and then choose **My Account**.
3. On the **Account Settings** page, scroll down to the section **AWS Regions**.

   Note
   You might be prompted to approve your access to this information. AWS sends a request to the email address associated with the account and to the primary contact phone number. Choose the link in the request to open it in your browser, and approve the access.
4. Next to each AWS Region with an option in the **Action** column, choose either **Enable** or **Disable**, depending on whether you want the users in your account to be able to create and access resources in that Region.
5. If prompted, confirm your choice.
6. After you have made all of your changes, choose **Update**.

AWS CLI & SDKs

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

Setting or changing your AWS account alias

The AWS account root user and AWS Identity and Access Management (IAM) users in the account sign in using a web URL.

If you want the URL for your IAM users to contain your company name (or another easy-to-remember identifier) instead of the AWS account ID, you can create an *account alias*. This section provides information about AWS account aliases and lists the API operations that you can use to create an alias.

The sign-in page URL for your account's IAM users has the following format, by default.

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

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

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

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

Tip
To create a bookmark for your account sign-in page in your web browser, we recommend that you manually type the sign-in URL in the bookmark entry. Don't use your web browser's "bookmark this page" feature, because that can capture a lot of information that relates only to your current browser session. That information can interfere with future visits to the page.

Considerations

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

- The account alias must be unique across all Amazon Web Services products. It must contain only lowercase letters, digits, and hyphens.

Creating, deleting, and viewing the AWS account alias

AWS Management Console

Creating or editing an account alias

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- `iam:ListAccountAliases`
- `iam:CreateAccountAlias`

1. Sign in to the AWS Management Console as either the AWS account root user or as an IAM user or role that has the minimum permissions.
2. Open the IAM console at https://console.aws.amazon.com/iam/.
3. In the navigation pane, choose **Dashboard**.
4. In the right-hand pane under **AWS account**, for **Sign-in URL for IAM users in this account**, choose **Customize**. If an alias already exists, then choose **Edit**.
5. For **Preferred alias**, enter the name you want to use for your alias, then choose **Save changes**.

   Note
   You can have only one alias associated with your AWS account at a time. If you create a new alias, the previous alias is removed, and the sign-in URL that was associated with the previous alias stops working.

Deleting an account alias

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- `iam:ListAccountAliases`
- `iam:CreateAccountAlias`
- `iam:DeleteAccountAlias`

1. Sign in to the AWS Management Console as either the AWS account root user or as an IAM user or role that has the minimum permissions.
2. Open the IAM console at https://console.aws.amazon.com/iam/.
3. In the navigation pane, choose **Dashboard**.
4. In the right-hand pane under **AWS account**, for **Sign-in URL for IAM users in this account**, choose **Delete**.

AWS CLI & SDKs

You can create, update, or delete an account alias by using the following SDK API operations or their AWS CLI equivalents:

- ListAccountAliases

```
aws iam list-account-aliases
```

- CreateAccountAlias

  `aws iam create-account-alias`
- DeleteAccountAlias

  `aws iam delete-account-alias`

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- `iam:ListAccountAliases`
- `iam:CreateAccountAlias`
- `iam:DeleteAccountAlias`

To display your AWS account ID alias, run the following command.

```
$ aws iam list-account-aliases
{
    "AccountAliases": [
        "myaccountalias"
    ]
}
```

To create an alias for your AWS Management Console sign-in, run the following command:

```
$ aws iam create-account-alias \
    --account-alias myaliasname
```

This command produces no output if it's successful.

To delete an AWS account ID alias, run the following command.

```
$ aws iam delete-account-alias \
--account-alias bisdavid
```

This command produces no output if it's successful.

Closing your AWS account

In this guide, we exclusively use the term *closing* an AWS account rather than *deleting* an account. Some elements of an AWS account are unique across all accounts, and using the term "delete" might mistakenly imply that some of those elements are available for future re-use. As an example, an account ID for a closed AWS account is ***never*** reused for obvious security reasons.

Only the AWS account root user can close an AWS account. AWS can't close an account on your behalf. If you have any questions about the process, contact your account representative or contact AWS Support for assistance. For more information about contacting AWS Support, see Contacting AWS Support.

Important
An AWS account ID number is ***never*** reused after its account is closed. This helps prevent the security risks that would otherwise occur. For example, consider if an account ID number was used in an AWS Identity and Access Management (IAM) permission policy and the ID suddenly referenced a different, and unexpected, AWS account.

Topics

Considerations before you close your AWS account

Before closing your AWS account, consider the following:

Your agreement with AWS

Closing your AWS account serves as notice to us that you want to cancel the AWS customer agreement or other agreement with AWS that governs your AWS account, solely with respect to this specific AWS account. If you reopen your AWS account during the *post-closure period* (within 90 days after you close the account), you agree that the same agreement terms govern your access to and use of the service offerings through your reopened AWS account.

AWS Management Console access

Your access to the AWS Management Console for a closed AWS account is restricted. During the post-closure period, you can still sign in to your AWS account to view your past billing information and access AWS Support. You can't access any other AWS services or start any new AWS services in the closed account.

Existing content and services still in use

After the post-closure period, AWS automatically deletes any remaining content in your AWS account, and terminates any AWS services that are still in use. Before closing your account, you should retrieve all content from the account. For instructions on how to retrieve your content, see the documentation for that service. For more information about the post-closure period, see Accessing your AWS account after you close it (p. 31).

Your payment method

We charge you through your designated payment method for any usage fees incurred before you closed your AWS account. We issue you any refunds that might be due through that same payment method. If you have active subscriptions (such as a Reserved Instance that you pay for monthly), even after your account is closed, you might continue to be charged for the subscription through your designated payment method until the subscription expires or is sold according to the terms governing the subscription. These charges and refunds might occur after you close your account.

In addition, if you reopen your account, you might be charged for the cost of running AWS services (that you didn't stop before closing your account) during the post-closure period. Closing your AWS account doesn't affect payment methods that you use on Amazon.com or other Amazon websites.

On-Demand charges

During the post-closure period, billing for On-Demand charges stops. However, you're billed for any usage that has accrued up until the time you closed your account. You'll be charged for that usage at the beginning of the next month. In addition, if you purchased any subscriptions with ongoing payment obligations, you might continue to be charged for them after your account is closed.

> **Important**
> You will continue to generate costs if you don't stop or delete your resources.

Domains registered with Amazon Route 53

Domains that are registered with Route 53 are not deleted automatically. When you close your AWS account, you have three options:

- You can disable automatic renewal, and the domains are automatically deleted when the registration period expires. For more information, see Enabling or Disabling Automatic Renewal for a Domain in the *Amazon Route 53 Developer Guide*.
- You can transfer the domains to another AWS account. For more information, see Transferring a Domain to a Different AWS account.
- You can transfer the domains to another domain registrar. For more information, see Transferring a Domain from Route 53 to Another Registrar.

If you already closed the account, you can open a case with AWS Support to get help with disabling automatic renewal or transferring your domains. For more information, see Contacting AWS Support About Domain Registration Issues. There is no charge to open a case for domain registration issues.

Charges if you reopen your AWS account

If you reopen your AWS account during the post-closure period, you might be billed for the cost of any AWS services that aren't stopped or resources that aren't deleted before you close your account.

Example

You reopen your AWS account 30 days after closure. Your AWS account had only an active `t2.micro` Amazon EC2 instance at closure. For this example, imagine that the price for a `t2.micro` Amazon EC2 instance in your AWS Region is $0.01 per hour. In this case, you might be charged for 30 days x 24 hours x $0.01 per hour = $7.20 for your AWS services.

Closing a member account

When you close an account that you created with AWS Organizations, that account isn't removed from the organization until after the post-closure period. During the post-closure period, a closed member account still counts toward your quota of accounts in the organization.

To avoid having the account count against the quota, remove member accounts from the organization before closing it. For more information, see Closing an AWS account in the *AWS Organizations User Guide*.

Cross-account access to the account you’re closing

After you close your AWS account, any access requests to your closed account's AWS services from other AWS accounts fail. This occurs even if you have granted the other accounts permission to access your account's AWS services. If you reopen your AWS account, other AWS accounts can again access your account's AWS services and resources if you granted the necessary permissions to the other AWS accounts.

Removing Amazon VPC peering connection

AWS doesn't delete Amazon Virtual Private Cloud (Amazon VPC) peering connections when you close one of the accounts participating in the VPC peering connection. Any traffic destined for the VPC peering connection originating from other active accounts is dropped because AWS terminates instances and deletes any security groups in the closed account. To remove the VPC peering connection, delete it from your account using the Amazon VPC console, AWS Command Line Interface (AWS CLI), or Amazon EC2 API. For more information, see Deleting a VPC peering connection

Troubleshooting errors when closing an AWS account

If you receive an error message while trying to close your AWS account, you can contact your account representative or contact AWS Support to open a billing or account support case for assistance. Common reasons why you might not be able to close your AWS account include the following:

- Your account is the management account of an organization in AWS Organizations with active member accounts. To close the management account, you must first remove all member accounts from the organization.
- You have unpaid invoices for your account.
- You are not signed in to the account as the AWS account root user.
- You are an active AWS Marketplace seller.

Closing your AWS account

You can close your AWS account using the following procedure.

AWS Management Console

To close your AWS account

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- You must sign in as the AWS account root user, which requires no additional IAM permissions. You can't perform these steps as an IAM user or role.

1. Sign in as the root user of the account that you want to close, using the email address and password that are associated with the account. If you sign in as an AWS Identity and Access Management (IAM) user or role, you can't close an account.
2. Open the Billing and Cost Management console at https://console.aws.amazon.com/billing/home#/.
3. On the navigation bar in the upper-right corner, choose your account name (or alias), and then choose **My Account**.
4. On the **Account Settings** page, scroll to the end of the page to the **Close Account** section. Read and ensure that you understand the text next to the check boxes. After you close an AWS account, you can no longer use it to access AWS services.
5. Select the check boxes to accept the terms, and then choose **Close Account**.
6. In the confirmation box, choose **Close Account**.

AWS CLI & SDKs

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

Accessing your AWS account after you close it

After you close an AWS account, you can no longer use it to access AWS services. However, for the 90 days after you close your account (known as the *Post-Closure Period*), you can view the past billing information for your AWS accountand access AWS Support.

During the Post-Closure Period, AWS may retain any content that you didn't delete and any AWS services that you didn't stop before you closed your AWS account. You can access any remaining content or AWS services only by reopening your account during the Post-Closure Period.

You can reopen your AWS account by contacting AWS Support. If you choose to reopen your account, you can access the content that you didn't delete and AWS services that you didn't stop before closing your account, but you might be charged for the cost of running those AWS services during the Post-Closure Period. You can estimate the cost of running AWS services by using the AWS Pricing Calculator in the *AWS Pricing Calculator User Guide*.

After the Post-Closure Period

After the Post-Closure Period, AWS permanently closes your AWS account, and you can no longer reopen it. Any content that you didn't delete is permanently deleted, and any AWS services that you didn't stop are stopped. Service attributes can be retained as long as needed for billing and administration purposes.

You can't create a new AWS account using the same alias or email address that was registered to your AWS account at the time of its closure.

Billing for your AWS account

For billing related procedures and tasks that are related to your AWS account, see the following topics in the *AWS Billing and Cost Management User Guide*:

- Changing which currency you use to pay your bill
- Updating and deleting tax registration numbers
- Enabling tax setting inheritance

Managing accounts in India

If you sign up for a new AWS account and choose India for your contact address, your user agreement is with Amazon Internet Services Private Limited (AISPL), a local AWS seller in India.AISPL manages your billing, and your invoice total is listed in Indian rupees (INR) instead of US dollars (USD). After you create an account with AISPL, you can't change the country in your contact information.

If you have an existing AWS account with an India address, your account is either with AWS or AISPL, depending on when you opened the account. To learn whether your account is with AWS or AISPL, see Determining which company your account is with (p. 33). If you're an existing AWS customer, you can continue to use your AWS account. You also can choose to have both an AWS account and an AISPL account, although they can't be consolidated into the same AWS organization. For information about managing an AWS account, see Managing your AWS account (p. 7).

If your account is with AISPL, follow the procedures in this topic to manage your account. This topic explains how to sign up for an AISPL account, edit information about your AISPL account, and add or edit your Permanent Account Number (PAN).

As part of the credit card verification during signup, AISPL charges your credit card 2 INR. AISPL refunds the 2 INR after verification is done. You might be redirected to your bank as part of the verification process.

Topics

Determining which company your account is with

AWS services are provided by both AWS and AISPL. Use this procedure to determine which seller your account is with.

AWS Management Console

To determine which company your account is with

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- This procedure requires no special permissions.

1. Open the AWS Management Console at AWS Management Console.
2. In the page footer at the bottom of the page, look at the copyright notice. If the copyright is for Amazon Web Services, then your account is with AWS. If the copyright is for Amazon Internet Services Private Ltd., then your account is with AISPL.

AWS CLI & SDKs

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

Creating an AWS account with AISPL

AISPL is a local seller of AWS in India. Use the following procedure to sign up for an AISPL account if your contact address is in India.

AWS Management Console

To sign up for an AISPL account

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- Because this operation occurs before you have an AWS account, this operation requires no AWS permissions.

1. Open the AWS Management Console, and then choose **Sign In to the Console**.
2. On the **Sign In** page, enter the email address that you want to use.
3. Under your email address, select **I am a new user**, and then choose **Sign in using our secure server**.
4. For each of the login credential fields, enter your information, and then choose **Create account**.
5. For each of the contact information fields, enter your information.
6. After you have read the customer agreement, select the terms and conditions check box, and then choose **Create Account and Continue**.
7. On the **Payment Information** page, enter the payment method that you want to use.

8. Under **PAN Information**, choose **No** if you don't have a Permanent Account Number (PAN) or want to add it later. If you have a PAN and want to add it now, choose **Yes**, and in the **PAN** field enter your PAN.
9. Choose **Verify Card and Continue**. You must provide your CVV as part of the verification process. AISPL charges your card 2 INR as part of the verification process. AISPL refunds the 2 INR after verification is done.
10. For **Provide a telephone number**, enter your phone number. If you have a phone extension, for **Ext**, enter your phone extension.
11. Choose **Call Me Now**. After a few moments, a four-digit pin will appear on your screen.
12. Accept the automated call from AISPL. On your phone keypad, enter the four-digit pin displayed on your screen.
13. Once the automated call verifies your contact number, choose **Continue to Select Your Support Plan**.
14. On the **Support Plan** page, select your support plan, and then choose **Continue**. After your payment method is verified and your account is activated, you receive an email message confirming the activation of your account.

AWS CLI & SDKs

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

Managing your AISPL account

Except for the following tasks, the procedures for managing your account are the same as accounts created outside of India. See Managing your AWS account (p. 7).

Use the AWS Management Console to perform the following tasks:

- Add or edit a Permanent Account Number (PAN)
- Edit multiple Permanent Account Numbers (PANs)
- Edit multiple Goods and Services Tax Numbers (GSTs)
- View a tax invoice

Using the AWS account root user

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

Important
We strongly recommend that you don't use the root user for your everyday tasks, including the administrative tasks. Instead, follow the best practice of using the root user only to create your first IAM user (p. 46). Then securely lock away the root user credentials and use them to perform only those few account and service management tasks that require you to sign in as the root user. For the list of those tasks, see Tasks that require root user credentials (p. 40). For a tutorial on how to set up an administrator for daily use, see Creating your first IAM admin user and user group in the *IAM User Guide*.

You can change the root user password, and create, rotate, deactivate, or delete access keys (access key IDs and secret access keys) for your root user. Anyone who has root user credentials for your AWS account has unrestricted access to all the resources in your account, including billing information.

You can change the email address and password on the Security Credentials page. You can also choose **Forgot password?** on the AWS sign-in page to reset your password.

Topics in this section

Signing in as the AWS account root user

We recommend that you sign in to your AWS account as the root user only when required to perform tasks that can be performed only by the root user (p. 40).

AWS Management Console

To sign in as the root user, navigate to the console sign-in page.

If you see text boxes asking for **Account ID**, **IAM user name**, and **Password**, then you previously signed in to the console with *IAM user* credentials. Your browser might remember this preference and open this account-specific sign-in page every time that you try to sign in. You can't use this version of the sign-in page to sign in as the root user. If you see the IAM user version of the sign-in page, choose **Sign in using root user email** at the bottom of the page to return to the main sign-in page. From there, you can choose to sign in as the root user using your AWS account email address and password.

If multi-factor authentication (MFA) is activated for your account's root user, then you are next prompted to enter the one-time password from your device.

AWS CLI & SDKs

To use the AWS CLI or run API operations from an SDK as the account's root user, you must first have credentials in the form of an access key pair. You can then use those in your AWS CLI or SDK profile (such as Python and Boto3) to authenticate your requests.

Warning
We strongly recommend, that as a best practice, that you do **not** create access key pairs for your root user. Only a few tasks require the root user (p. 40), and you typically perform those tasks infrequently enough that we recommend signing into the AWS Management Console and performing the tasks there.

To create access keys for the root user, see Creating and deleting access keys for the AWS account root user (p. 37).

Activate MFA on the AWS account root user

To enhance the security of your root user credentials, we recommend that you follow the security best practice to activate multi-factor authentication (MFA) for your AWS account. Because the root user can perform sensitive operations in your account, adding this additional layer of authentication helps you to better secure your account. Multiple types of MFA are available.

For more information about MFA in the AWS environment, see Multi-factor authentication and the following topics in the *IAM User Guide*:

- To resynchronize a virtual or hardware MFA device
- To deactivate an MFA device
- What if an MFA device is lost or stops working?

You can activate **one** MFA device (of any kind) per root user or IAM user.

You activate an MFA device by using the AWS Identity and Access Management (IAM) console. Based on the type of MFA device you want to activate, choose one of the following topics in the *IAM User Guide*:

- Enable a virtual MFA device for your AWS account root user
- Enable a U2F security key for your AWS account root user
- Enable a hardware MFA device for your AWS account root user

Changing the password for the root user

To change the root user's password, you must sign in as the AWS account root user and not as an IAM user. To learn how to reset a *forgotten* root user password, see Resetting lost or forgotten passwords or access keys for AWS in the *IAM User Guide*.

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

- Change your password periodically.
- Keep your password private because anyone who knows your password can access your account.
- Use a different password on AWS than you use on other sites.
- Avoid passwords that are easy to guess. These include passwords such as `secret`, `password`, `amazon`, or `123456`. Also avoid things like dictionary words, your name, email address, or other personal information that someone can easily obtain.

AWS Management Console

To change the password for the root user

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- You must sign in as the AWS account root user, which requires no additional AWS Identity and Access Management (IAM) permissions. You can't perform these steps as an IAM user or role.

1. Use your AWS account's email address and password to sign in to the AWS Management Console as your AWS account root user.
2. In the upper right corner of the console, choose your account name or number and then choose **My Account**.
3. On the right side of the page, in the **Account Settings** section, choose **Edit**.
4. On the **Password** line, choose **Edit** to change your password.
5. Choose a strong password. Although you can set an account password policy for IAM users, that policy doesn't apply to the root user.

   AWS requires that your password meet the following conditions:

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

   Note
   AWS is rolling out improvements to the sign-in process. One of those improvements is to enforce a more secure password policy for your account. If AWS has upgraded your account, you are required to meet the password policy described earlier. If AWS hasn't yet upgraded your account, then AWS doesn't yet enforce this policy. However, we strongly recommend that you follow its guidelines for a more secure password.

AWS CLI & SDKs

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

Creating and deleting access keys for the AWS account root user

Although we don't recommend it (p. 46), you can create access keys for your root user so that you can run commands in the AWS Command Line Interface (AWS CLI) or use API operations from one of the AWS SDKs.

Creating access keys for the root user

You can use the AWS Management Console or AWS programming tools to create access keys for the root user.

AWS Management Console

To create an access key for the AWS account root user

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- You must sign in as the AWS account root user, which requires no additional AWS Identity and Access Management (IAM) permissions. You can't perform these steps as an IAM user or role.

1. Use your AWS account's email address and password to sign in to the AWS Management Console as your AWS account root user.
2. Choose your account name in the navigation bar, and then choose **My Security Credentials**.
3. If you see a warning about accessing the security credentials for your AWS account, choose **Continue to Security Credentials**.
4. Expand the **Access keys (access key ID and secret access key)** section.
5. Choose **Create New Access Key**. If this option is not available, then you already have the maximum number of access keys. You must delete one of the existing access keys before you can create a new key. For more information, see IAM Object Quotas in the *IAM User Guide*.

   A warning explains that you have only this one opportunity to view or download the secret access key. You can't retrieve it later.

   - If you choose **Show Access Key**, you can copy the access key ID and secret key from your browser window and paste it somewhere else.
   - If you choose **Download Key File**, you receive a file named `rootkey.csv` that contains the access key ID and the secret key. Save the file somewhere safe.
6. When you no longer need the access key we recommend that you delete it (p. 38), or at least mark it inactive by choosing **Make Inactive** so that no one can misuse it.

AWS CLI & SDKs

To create an access key for the root user

Note
To run the following command or API operation as the root user, you must already have one active access key pair. If you don't have any access keys, create the first access key using the AWS Management Console. Then, you can use the credentials from that first access key with the AWS CLI to create the second access key, or to delete an access key.

- AWS CLI: aws iam create-access-key

  Example

```
$ aws iam create-access-key
{
    "AccessKey": {
        "UserName": "MyUserName",
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "Status": "Active",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "CreateDate": "2021-04-08T19:30:16+00:00"
    }
}
```

- AWS API: CreateAccessKey

Deleting access keys for the root user

You can use the AWS Management Console to delete access keys for the root user. You can't use the AWS CLI or the AWS API to delete the root user access keys.

AWS Management Console

To delete an access key for the root user

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- You must sign in as the AWS account root user, which requires no additional AWS Identity and Access Management (IAM) permissions. You can't perform these steps as an IAM user or role.

1. Use your AWS account's email address and password to sign in to the AWS Management Console as your AWS account root user.
2. Choose your account name in the navigation bar, and then choose **My Security Credentials**.
3. If you see a warning about accessing the security credentials for your AWS account, choose **Continue to Security Credentials**.
4. Expand the **Access keys (access key ID and secret access key)** section.
5. Find the access key that you want to delete, and then, under the **Actions** column, choose **Delete**.

   Note
   You can mark an access key as inactive instead of deleting it. This way you can resume using it in the future without having to change either the key ID or secret key. While the key is inactive, any attempts to use it in requests to the AWS API fail with the error access denied.

AWS CLI & SDKs

To delete an access key for the root user

Minimum permissions
To perform the following steps, you must have at least the following IAM permissions:

- You must sign in as the AWS account root user, which requires no additional AWS Identity and Access Management (IAM) permissions. You can't perform these steps as an IAM user or role.

- AWS CLI: aws iam delete-access-key

  Example

  ```
  $ aws iam delete-access-key \
      --access-key-id AKIAIOSFODNN7EXAMPLE
  ```

  This command produces no output when successful.
- AWS API: DeleteAccessKey

Comparing AWS account root user credentials and IAM user credentials

There are two different types of users in AWS. You are either the account owner (root user) or you are an AWS Identity and Access Management (IAM) user. The root user is created when the AWS account is created. IAM users are created by the root user or an IAM administrator for the account. All AWS users have security credentials.

Root user credentials

The credentials of the account owner allow full access to all resources in the account. You can't use IAM policies to explicitly deny the root user access to resources. You can only use an AWS Organizations service control policy (SCP) to limit the permissions of the root user of a member account. Because of this, we recommend that you create an IAM user with administrator permissions to use for everyday AWS tasks and lock away the access keys for the root user.

There are specific tasks that can be performed only by the root user. For example, only the root user can close your account. If you need to perform a task that requires the root user, sign in to the AWS Management Console using the email address and password of the root user. For more information, see Tasks that require root user credentials (p. 40).

IAM credentials

With IAM, you can securely control access to AWS services and resources for users in your AWS account. For example, if you require administrator-level permissions, you can create an IAM user, grant that user full access to your account, and then use those credentials to interact with AWS. If you need to modify or revoke your permissions, you can delete or modify the policies that are associated with that IAM user.

If you have multiple users that require access to your AWS account, create unique credentials for each user and define who has access to which resources. You don't need to and ***shouldn't*** share credentials. For example, you can create IAM users with read-only access to resources in your AWS account and distribute the credentials for each IAM user to one of your users.

Tasks that require root user credentials

We recommend that you use an IAM user with appropriate permissions to perform tasks and access AWS resources. However, you can perform the tasks listed below only when you sign in as the root user of an account.

Tasks

- Change your account settings. This includes the account name, email address, root user password, and root user access keys. Other account settings, such as contact information, payment currency preference, and AWS Regions, don't require root user credentials.
- Restore IAM user permissions. If the only IAM administrator accidentally revokes their own permissions, you can sign in as the root user to edit policies and restore those permissions.
- Activate IAM access to the Billing and Cost Management console.
- View certain tax invoices. An IAM user with the aws-portal:ViewBilling permission can view and download VAT invoices from AWS Europe, but not AWS Inc. or Amazon Internet Services Private Limited (AISPL).
- Close your AWS account.
- Change your AWS Support plan or Cancel your AWS Support plan. For more information, see IAM for AWS Support.
- Register as a seller in the Reserved Instance Marketplace.
- Configure an Amazon S3 bucket to enable MFA (multi-factor authentication).
- Edit or delete an Amazon Simple Storage Service (Amazon S3) bucket policy that includes an invalid virtual private cloud (VPC) ID or VPC endpoint ID.
- Sign up for GovCloud.

Using AWS Account Management in your organization

AWS Organizations is an AWS service that you can use to manage your AWS accounts as a group. This provides features like consolidated billing, where all of your accounts' bills are grouped together and handled by a single payer. You can also centrally manage the security of your organization by using policy based controls.

For more information about AWS Organizations, see the AWS Organizations User Guide.

AWS Account Management extends that functionality to also include the management of the metadata attached to an AWS account, such as the alternate contact information.

When you use AWS Organizations to manage your accounts as a group, most administrative tasks for the organization can be performed by only the organization's *management account*. By default, this includes only operations related to managing the organization itself. You can extend this additional functionality to other AWS services by enabling *trusted access* between Organizations and that service. Trusted access grants permissions to the specified AWS service to access information about the organization and the accounts it contains. When you enable trusted access for Account Management, the Account Management service grants Organizations and its management account permissions to access the metadata for all of the organization's member accounts.

After you enable trusted access, you can also choose to designate one of your member accounts as a *delegated admin* account for AWS Account Management. This allows the delegated admin account to perform the same Account Management metadata management tasks for the member accounts in your organization that previously only the management account could do. The delegated admin account can access only the management tasks for the Account Management service. The delegated admin account doesn't have all of the administrative access to the organization that the management account has.

Enabling trusted access and delegated admin result in you being able to use the `accountID` parameter in those Account Management API operations (p. 76) that support it. You can use this parameter successfully only if you call the operation using credentials from the management account if you enabled trusted access, or from the Account Management delegated admin account if you enable one.

When your AWS account is part of an organization that is managed by AWS Organizations, then the administrator of the organization can apply service control policies (SCPs) that can limit what the principals in member accounts can do. An SCP never grants permissions; instead, it is a filter that limits what permissions can be used by the member account. A user or role (*a principal*) in a member account can perform only those operations that are in the intersection of what is allowed by the SCPs that apply to the account and the IAM permission policies attached to the principal. For example, you can use SCPs to prevent any principal in an account from modifying their own account's alternate contacts. For example SCPs that apply to AWS accounts, see Restricting access with AWS Organizations service control policies (p. 43).

Topics

Enabling trusted access for AWS Account Management

To enable the management account in your organization to call the AWS Account Management API operations for other member accounts in the organization, use the following procedure.

Minimum permissions
To perform these tasks, you must meet the following requirements:

- You can perform this only from the organization's management account.
- Your organization must have all features enabled.

AWS Management Console

To enable trusted access with AWS Organizations for AWS Account Management

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. Choose **Services** in the navigation pane.
3. Choose **AWS Account Management** in the list of services.
4. Choose **Enable trusted access**.

AWS CLI & SDKs

To enable trusted access with AWS Organizations for AWS Account Management

You can use the following commands to enable trusted access for Account Management in your organization.

- AWS CLI: enable-aws-service-access

  The following example enables trusted access for AWS Account Management in the calling account's organization.

```
$ aws organizations enable-aws-service-access \
    --service-principal account.amazonaws.com
```

  This command produces no output if it's successful.

  After running this command, you can use credentials from the organization's management account to call Account Management API operations that use the `--accountId` parameter to reference member accounts in an organization.

Enabling a delegated admin account for AWS Account Management

A delegated admin account can call the AWS Account Management API operations for other member accounts in the organization. To designate a member account in your organization as a delegated admin account, use the following procedure.

Minimum permissions
To perform these tasks, you must meet the following requirements:

- You can perform this only from the organization's management account.
- Your organization must have all features enabled.
- You must have enabled trusted access for Account Management in your organization (p. 42).

After you specify a delegated admin account for your organization, users and roles in that account can call the AWS CLI and AWS SDK operations in the `account` namespace that can work in the Organizations mode by supporting an optional `AccountId` parameter.

AWS Management Console

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

AWS CLI & SDKs

To register a delegated admin account for the Account Management service

You can use the following commands to enable a delegated admin for the Account Management service.

You must specify the following service principal:

```
account.amazonaws.com
```

- AWS CLI: register-delegated-administrator

  The following example registers a member account of the organization as a delegated admin for the Account Management service.

  ```
  $ aws organizations register-delegated-administrator \
      --account-id 123456789012 \
      --service-principal account.amazonaws.com
  ```

  This command produces no output if it's successful.

  After you run this command, you can use credentials from account 123456789012 to call Account Management AWS CLI and SDK API operations that use the `--account-id` parameter to reference member accounts in an organization.

Restricting access with AWS Organizations service control policies

This topic presents examples that show how you can use service control policies (SCPs) to restrict what the users and roles in the accounts in your organization can do. For more information about service control policies, see the following topics in the *AWS Organizations User Guide*:

- Creating SCPs
- Attaching SCPs to OUs and accounts
- Strategies for SCPs

- SCP policy syntax

Example Example 1: Prevent accounts from modifying their own alternate contacts

The following example denies the `PutAlternateContact` and `DeleteAlternateContact` API operations from being called by any member account in standalone account mode (p. 13). This prevents any principal in the affected accounts from changing their own alternate contacts.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Deny",
            "Action": [
                "account:PutAlternateContact",
                "account:DeleteAlternateContact"
            ],
            "Resource": [ "arn:aws:account::*:account" ]
        }
    ]
}
```

Example Example 2: Prevent any member account from modifying alternate contacts for any other member account in the organization

The following example generalizes the `Resource` element to "*", which means that it applies to both standalone mode requests and organizations mode requests (p. 13). This means that even the delegated admin account for Account Management, if the SCP applies to it, is blocked from changing any alternate contact for any account in the organization.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Deny",
            "Action": [
                "account:PutAlternateContact",
                "account:DeleteAlternateContact"
            ],
            "Resource": [ "*" ]
        }
    ]
}
```

Example Example 3: Prevent a member account in an OU from modifying its own alternate contacts

The following example SCP includes a condition that compares the account's organization path to a list of two OUs. This results in blocking a principal in any account in the specified OUs from modifying their own alternate contacts.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Deny",
            "Action": "account:PutAlternateContact",
            "Resource": [
                "arn:aws:account::*:account"
            ],
            "Condition": {
                "ForAnyValue:StringLike": {
                    "account:AccountResourceOrgPath": [
                        "o-aa111bb222/r-a1b2/ou-a1b2-f6g7h111/",
                        "o-aa111bb222/r-a1b2/ou-a1b2-f6g7h222/"
                    ]
                }
            }
    ]
}
```

Best practices for AWS accounts

This section describes best practices that we recommend that you follow with your AWS accounts. They are arranged in the following categories:

Topics

Best practices to protect your account's root user

The following are recommended best practices related to the root user of an AWS account.

Limit the tasks you perform with the root user

We strongly recommend that you use the root user only for two things:

- Create the first administrator user in AWS Identity and Access Management (IAM). For details about how to do this, see Creating your first IAM user and group in the *IAM User Guide*.
- Perform those tasks that can be performed by ***only*** the root user. For the complete list of these tasks, see Tasks that require root user credentials (p. 40).

Lock away your AWS account root user access keys

You use an access key (an access key ID and secret access key) to make programmatic requests to AWS. However, we strongly recommend that you do not use your AWS account root user access key. The access key for your AWS account root user gives full access to all your resources for all AWS services, including your billing information. You can't reduce the permissions associated with your AWS account root user access key.

You should protect your root user credentials like you would your credit card numbers or any other sensitive secret. Here are some ways to do that:

- Access keys
  - If you don't already have an access key for your AWS account root user, don't create one unless you absolutely need to. Instead, use the root user to create an IAM user for yourself that has administrative permissions.
  - If you do have an access key for your root user, delete it.
  - If you must keep one available, rotate (change) the access key regularly. To delete or rotate your root user access keys, use your root user to sign in to the My Security Credentials page in the AWS Management Console. You can manage your access keys in the **Access keys** section. For more information about rotating access keys, see Rotating access keys in the *IAM User Guide*.
- Never share your AWS account root user password or access keys with anyone.
- Use a strong password to help protect access to the AWS Management Console. For information about managing your AWS account root user password, see Changing the password for the root user (p. 36).
- Enable AWS multi-factor authentication (MFA) on your AWS account root user account. For more information, see Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

Best practices for managing AWS access keys

When you use AWS programmatically, you provide your AWS access keys so that AWS can verify your identity in programmatic calls. Your access keys consist of an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`).

Anyone who has your access keys has the same level of access to your AWS resources that you do. Consequently, AWS goes to significant lengths to protect your access keys, and, in keeping with our shared-responsibility model, you should as well.

The steps that follow can help you protect your access keys. For background information, see Creating and deleting access keys for the AWS account root user (p. 37).

Note
Your organization may have different security requirements and policies than those described in this topic. The suggestions provided here are intended as general guidelines.

Remove (or don't generate) an account access key

You must use your access keys to sign requests that you make using the AWS Command Line Tools, the AWS SDKs, or direct API calls. Anyone who has the access keys for your AWS account root user has unrestricted access to all the resources in your account, including billing information. You can't restrict the permissions for your AWS account root user.

One of the best ways to protect your account is to not have access keys for your AWS account root user. Unless you must have root user access keys (which is rare), it is best not to generate them. Instead, the recommended best practice is to create one or more AWS Identity and Access Management (IAM) users. Grant those IAM users the necessary permissions, and use them for everyday interaction with AWS.

If you already have access keys for your account, we recommend the following: Find places in your applications where you are currently using access keys (if any), and replace the root user access keys with IAM user access keys. Then disable and remove the root user access keys. For more information about how to substitute one access key for another, see How to Rotate Access Keys for IAM Users on the AWS Security Blog.

By default, AWS doesn't generate access keys for new accounts.

For information about how to create an IAM user with administrative permissions, see Creating Your First IAM Admin User and Group in the *IAM User Guide*.

Use temporary security credentials (IAM roles) instead of long-term access keys

In many scenarios, you don't need long-term access keys that never expire (as you have with an IAM user). Instead, you can create IAM roles and generate temporary security credentials. Temporary security credentials consist of an access key ID and a secret access key, but they also include a security token that indicates when the credentials expire.

Long-term access keys, such as those associated with IAM users and AWS account root users, remain valid until you manually revoke them. However, temporary security credentials obtained through IAM roles and other features of the AWS Security Token Service expire after a short period of time. Use temporary security credentials to help reduce your risk in case credentials are accidentally exposed.

Use an IAM role and temporary security credentials in these scenarios:

- **You have an application or AWS CLI scripts running on an Amazon EC2 instance.** Don't use access keys directly in your application. Don't pass access keys to the application, embed them in the application, or let the application read access keys from any source. Instead, define an IAM role that has appropriate permissions for your application and launch the Amazon Elastic Compute Cloud (Amazon EC2) instance with roles for EC2. Doing this associates an IAM role with the Amazon EC2 instance. This practice also enables the application to get temporary security credentials that it can in turn use to make programmatic calls to AWS. The AWS SDKs and the AWS Command Line Interface (AWS CLI) can get temporary credentials from the role automatically.

- **You need to grant cross-account access.** Use an IAM role to establish trust between accounts, and then grant users in one account limited permissions to access the trusted account. For more information, see Tutorial: Delegate access across AWS accounts using IAM roles in the *IAM User Guide*.
- **You have a mobile app.** Don't embed access keys with the app, even in encrypted storage. Instead, use Amazon Cognito to manage user identities in your app. This service lets you authenticate users using Login with Amazon, Facebook, Google, or any OpenID Connect (OIDC)–compatible identity provider. You can then use the Amazon Cognito credentials provider to manage credentials that your app uses to make requests to AWS. For more information, see Using the Amazon Cognito Credentials Provider on the AWS Mobile Blog.
- **You want to federate into AWS and your organization supports SAML 2.0.** If you work for an organization that has an identity provider that supports SAML 2.0, configure the provider to use SAML. You can use SAML to exchange authentication information with AWS and get back a set of temporary security credentials. For more information, see About SAML 2.0-based Federation in the *IAM User Guide*.
- **You want to federate into AWS and your organization has an on-premises identity store.** If users can authenticate inside your organization, you can write an application that can issue them temporary security credentials for access to AWS resources. For more information, see Enabling custom identity broker access to the AWS Management Console in the *IAM User Guide*.

Manage IAM user access keys properly

If you must create access keys for programmatic access to AWS, create them for IAM users, granting the users only the permissions they require. For more information, see Managing access keys for IAM users in the *IAM User Guide*.

> **Note**
> Are you using an Amazon EC2 instance with an application that requires programmatic access to AWS resources? If so, use IAM roles for EC2.

Observe these precautions when using access keys:

- **Don't embed access keys directly into code.** The AWS SDKs and the AWS Command Line Tools enable you to put access keys in known locations so that you don't have to keep them in code.

  Put access keys in one of the following locations:

  - **The AWS credentials file.** The AWS SDKs and AWS CLI automatically use the credentials that you store in the AWS credentials file.

    For information about using the AWS credentials file, see the documentation for your SDK. Examples include Set AWS Credentials and Region in the *AWS SDK for Java Developer Guide* and Configuration and credential files in the *AWS Command Line Interface User Guide*.

    To store credentials for the AWS SDK for .NET and the AWS Tools for Windows PowerShell, we recommend that you use the SDK Store. For more information, see Using the SDK Store in the *AWS SDK for .NET Developer Guide*.
  - **Environment variables.** On a multitenant system, choose user environment variables, not system environment variables.

    For more information about using environment variables to store credentials, see Environment Variables in the *AWS Command Line Interface User Guide*.

- **Use different access keys for different applications.** Do this so that you can isolate the permissions and revoke the access keys for individual applications if they are exposed. Having separate access keys for different applications also generates distinct entries in AWS CloudTrail log files. This configuration makes it easier for you to determine which application performed specific actions.
- **Rotate access keys periodically.** Change access keys on a regular basis. For details, see Rotating access keys (AWS CLI, Tools for Windows PowerShell, and AWS API) in the *IAM User Guide* and How to Rotate Access Keys for IAM Users on the AWS Security Blog.
- **Remove unused access keys.** If a user leaves your organization, remove the corresponding IAM user so that the user can no longer access your resources. To find out when an access key was last used, use the `GetAccessKeyLastUsed` API (AWS CLI command: `aws iam get-access-key-last-used`).
- **Configure multi-factor authentication for your most sensitive operations.** For more information, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.

Access the mobile app using AWS access keys

You can access a limited set of AWS services and features using the AWS mobile app. The mobile app helps you support incident response while on the go. For more information and to download the app, see AWS Console Mobile Application.

You can sign in to the mobile app using your console password or your access keys. As a best practice, do not use root user access keys. Instead, we strongly recommend that in addition to using a password or biometric lock on your mobile device, you create an IAM user to manage AWS resources. If you lose your mobile device, you can remove the IAM user's access. For more information about generating access keys for an IAM user, see Managing access keys for IAM users in the *IAM User Guide*.

To sign in using access keys (mobile app)

1. Open the app on your mobile device.
2. If this is the first time that you're adding an identity to the device, choose **Add an identity** and then choose **Access keys**.

   If you have already signed in using another identity, choose the menu icon and choose **Switch identity**. Then choose **Sign in as a different identity** and then **Access keys**.
3. On the **Access keys** page, enter your information:

   - **Access key ID** – Enter your access key ID.
   - **Secret access key** – Enter your secret access key.
   - **Identity name** – Enter the name of the identity that will appear in the mobile app. This does not need to match your IAM user name.
   - **Identity PIN** – Create a personal identification number (PIN) that you will use for future sign-ins.

     Note
     If you enable biometrics for the AWS mobile app, you will be prompted to use your fingerprint or facial recognition for verification instead of the PIN. If the biometrics fail, you might be prompted for the PIN instead.
4. Choose **Verify and add keys**.

   You can now access a select set of your resources using the mobile app.

Learn more

For more information about best practices for keeping your AWS account secure, see the following resources:

- IAM Best Practices. Contains suggestions for using the AWS Identity and Access Management (IAM) service to help secure your AWS resources
- The following topics provide guidance for setting up the AWS SDKs and the AWS CLI to use access keys:
  - Set AWS credentials and Region in the *AWS SDK for Java Developer Guide*
  - Using the SDK Store in the *AWS SDK for .NET Developer Guide*
  - Providing Credentials to the SDK in the *AWS SDK for PHP Developer Guide*
  - Configuration in the Boto 3 (AWS SDK for Python) documentation
  - Using AWS Credentials in the *AWS Tools for Windows PowerShell User Guide*
  - Configuration and credential files in the *AWS Command Line Interface User Guide*
  - Granting access using an IAM role in the *AWS SDK for .NET Developer Guide*. Discusses how programs written using the AWS SDK for .NET can automatically get temporary security credentials when running on an Amazon EC2 instance. Similar information is available for the AWS SDK for Java.

Security in AWS Account Management

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Account Management, see AWS services in scope by compliance program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations

This documentation helps you understand how to apply the shared responsibility model when using AWS Account Management. It shows you how to configure Account Management to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Account Management resources.

Topics

Data protection in AWS Account Management

The AWS shared responsibility model applies to data protection in AWS Account Management. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form fields such as a **Name** field. This includes when you work with Account Management or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

AWS PrivateLink for AWS Account Management

If you use Amazon Virtual Private Cloud (Amazon VPC) to host your AWS resources, you can access the AWS Account Management service from within the VPC without having to cross the public internet.

Amazon VPC lets you launch AWS resources in a custom virtual network. You can use a VPC to control your network settings, such as the IP address range, subnets, route tables, and network gateways. For more information about VPCs, see the *Amazon VPC User Guide*.

To connect your Amazon VPC to Account Management, you must first define an *interface VPC endpoint*, which lets you connect your VPC to other AWS services. The endpoint provides reliable, scalable connectivity, without requiring an internet gateway, network address translation (NAT) instance, or VPN connection. For more information, see Interface VPC Endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*.

Creating the Endpoint

You can create an AWS Account Management endpoint in your VPC using the AWS Management Console, the AWS Command Line Interface (AWS CLI), an AWS SDK, the AWS Account Management API, or AWS CloudFormation.

For information about creating and configuring an endpoint using the Amazon VPC console or the AWS CLI, see Creating an Interface Endpoint in the *Amazon VPC User Guide.*

Note

When you create an endpoint, specify Account Management as the service that you want your VPC to connect to, using the following format:

```
com.amazonaws.us-east-1.account
```

You must use the string exactly as shown, specifying the `us-east-1` Region. As a global service, Account Management is hosted in only that one AWS Region.

For information about creating and configuring an endpoint using AWS CloudFormation, see the AWS::EC2::VPCEndpoint resource in the *AWS CloudFormation User Guide*.

Amazon VPC Endpoint Policies

You can control what actions can performed through this service endpoint by attaching an endpoint policy when you create the Amazon VPC endpoint. You can create complex IAM rules by attaching multiple endpoint policies. For more information, see:

- Amazon Virtual Private Cloud endpoint policies for Account Management (p. 53)
- Controlling Access to Services with VPC Endpoints in the *AWS PrivateLink Guide*.

Amazon Virtual Private Cloud endpoint policies for Account Management

You can create a Amazon VPC endpoint policy for Account Management in which you specify the following:

- The principal that can perform actions.
- The actions that the principals can perform.
- The resources on which the actions can be performed.

The following example shows an Amazon VPC endpoint policy that allows one IAM user named Alice in account 123456789012 to both retrieve and change the alternate contact information for any AWS account, but denies all IAM users permission to delete any alternate contact information on any account.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "account:GetAlternateContact",
                "account:PutAlternateContact"
            ],
            "Resource": "arn:aws::iam:*:account,
            "Effect": "Allow",
            "Principal": {
              "AWS": "arn:aws::iam:123456789012:user/Alice"
            }
        },
        {
            "Action": "account:DeleteAlternateContact",
            "Resource": "*",
            "Effect": "Deny",
            "Principal": "arn:aws::iam:*:root"
        }
    ]
}
```

If you want to grant access to accounts that are part of an AWS Organization to a principal that is in one of the organization's member accounts, then the `Resource` element must use the following format:

```
arn:aws:account::{ManagementAccountId}:account/o-{OrganizationId}/{AccountId}
```

For more information about creating endpoint policies, see Controlling Access to Services with VPC Endpoints in the *AWS PrivateLink Guide*.

Identity and Access Management for AWS Account Management

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Account Management resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Account Management.

Service user – If you use the Account Management service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Account Management features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Account Management, see Troubleshooting AWS Account Management identity and access (p. 65).

Service administrator – If you're in charge of Account Management resources at your company, you probably have full access to Account Management. It's your job to determine which Account Management features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Account Management, see How AWS Account Management works with IAM (p. 58).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Account Management. To view example Account Management identity-based policies that you can use in IAM, see Identity-based policy examples for AWS Account Management (p. 63).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see Signing in to the AWS Management Console as an IAM user or root user in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email address or your IAM user name. You can access AWS programmatically using your root user or IAM users access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 signing process in the *AWS General Reference*.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *AWS General Reference*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see Managing access keys for IAM users in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated users and roles in the *IAM User Guide*.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role

as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, resources, and condition keys for AWS Account Management in the *Service Authorization Reference*.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

Every IAM entity (user or role) starts with no permissions. By default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS Account Management works with IAM

Before you use IAM to manage access to Account Management, learn what IAM features are available to use with Account Management.

IAM features you can use with AWS Account Management

IAM feature	Account Management support
Identity-based policies (p. 58)	Yes
Resource-based policies (p. 59)	No
Policy actions (p. 59)	Yes
Policy resources (p. 60)	Yes
Policy condition keys (p. 60)	Yes
ACLs (p. 62)	No
ABAC (tags in policies) (p. 62)	Yes
Temporary credentials (p. 62)	Yes
Principal permissions (p. 62)	Yes
Service roles (p. 63)	No
Service-linked roles (p. 63)	No

To get a high-level view of how Account Management and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for Account Management

Supports identity-based policies	Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for Account Management

To view examples of Account Management identity-based policies, see Identity-based policy examples for AWS Account Management (p. 63).

Resource-based policies within Account Management

Supports resource-based policies	No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Policy actions for Account Management

Supports policy actions	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Account Management actions, see Actions defined by AWS Account Management in the *Service Authorization Reference*.

Policy actions in Account Management use the following prefix before the action.

```
account
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "account:action1",
      "account:action2"
         ]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that work with an AWS account's alternate contacts, include the following action.

```
"Action": "account:*AlternateContact"
```

To view examples of Account Management identity-based policies, see Identity-based policy examples for AWS Account Management (p. 63).

Policy resources for Account Management

Supports policy resources	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

The Account Management service supports the following specific resource types in an IAM policy's `Resources` element to help you filter the policy and distinguish between these types of AWS accounts:

- **account**

  This `resource` type matches only standalone AWS accounts that are not member accounts in an organization managed by the AWS Organizations service.
- **accountInOrganization**

  This `resource` type matches only AWS accounts that are member accounts in an organization managed by the AWS Organizations service.

To see a list of Account Management resource types and their ARNs, see Resources defined by AWS Account Management in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by AWS Account Management.

To view examples of Account Management identity-based policies, see Identity-based policy examples for AWS Account Management (p. 63).

Policy condition keys for Account Management

Supports service-specific policy condition keys	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

The Account Management service supports the following condition keys that you can use to provide fine-grained filtering for your IAM policies:

- **account:TargetRegion**

  This condition key takes an argument that consists of a list of AWS Region codes. It lets you filter the policy to affect only those actions that apply to the specified Regions.
- **account:AlternateContactTypes**

  This condition key takes a list of alternate contact types:
  - BILLING
  - OPERATIONS
  - SECURITY

  Using this key lets you filter the request to only those actions that target the specified alternate contact types.
- **account:AccountResourceOrgPaths**

  This condition key takes an argument that consists of a list of ARNs with wildcards that represent accounts in an organization. It lets you filter the policy to affect only those actions that target accounts with ARNs that match. For example, the following ARN matches only those accounts in the specified organization and the specified organizational unit (OU).

  ```
  arn:aws:organizations::111111111111:ou/o-aa111bb222/ou-a1b2-f6g7h111/*
  ```
- **account:AccountResourceOrgTags**

  This condition key takes an argument that consists of a list of tag keys and values. It lets you filter the policy to affect only those accounts that are members of an organization and that are tagged with the specified tag keys and values.

To see a list of Account Management condition keys, see Condition keys for AWS Account Management in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by AWS Account Management.

To view examples of Account Management identity-based policies, see Identity-based policy examples for AWS Account Management (p. 63).

Access control lists in Account Management

Supports ACLs	No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Attribute-based access control with Account Management

Supports ABAC (tags in policies)	Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with Account Management

Supports temporary credentials	Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for Account Management

Supports principal permissions	Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, resources, and condition keys for AWS Account Management in the *Service Authorization Reference*.

Service roles for Account Management

Supports service roles	No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

Service-linked roles for Account Management

Supports service-linked roles	No

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a `Yes` in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for AWS Account Management

By default, users and roles don't have permission to create or modify Account Management resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform actions on the resources that they need. The administrator must then attach those policies for users that require them.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

Topics

Policy best practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete Account Management resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started using AWS managed policies** – To start using Account Management quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get started using permissions with AWS managed policies in the *IAM User Guide*.
- **Grant least privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant least privilege in the *IAM User Guide*.
- **Enable MFA for sensitive operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.
- **Use policy conditions for extra security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.

Using the Account Management Settings page in the AWS Management Console

To access the **Accounts Settings** page in the AWS Management Console, you must have a minimum set of permissions. These permissions must allow you to list and view details about your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (IAM users or roles) with that policy.

To ensure that users and roles can use the Account Management console, you can choose to attach the either the `AWSAccountManagementReadOnlyAccess` or `AWSAccountManagementFullAccess` AWS managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, in many cases you can choose to allow access to only the actions that match the API operations that you're trying to perform.

Providing read-only access to the Account Settings page in the AWS Management Console

In the following example, you want to grant an IAM user in your AWS account read-only access to the Account Settings page in the AWS Management Console. Users with this policy attached can't make any changes.

The `aws-portal:ViewAccount` grants access to view most of the settings on the **Account Settings** page. However, to view the currently enabled AWS Regions, you must also include the `account:ListRegions` action.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GrantReadOnlyAccessToAccountSettings",
            "Effect": "Allow",
            "Action": [
                "aws-portal:ViewAccount",
                "account:ListRegions"
            ],
            "Resource": "*"
        }
    ]
}
```

Providing full access to the Account Settings page in the AWS Management Console

In the following example, you want to grant an IAM user in your AWS account full access to the Account Settings page in the AWS Management Console. Users with this policy attached can alter settings for the account.

This example policy builds on the preceding example policy by adding the `aws-portal:ModifyAccount` permission, which allows the user to change most of the settings for the account, and also adds the `account:EnableRegion` and `account:DisableRegion` permissions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GrantFullAccessToAccountSettings",
            "Effect": "Allow",
            "Action": [
                "aws-portal:ViewAccount",
                "account:ListRegions",
                "aws-portal:ModifyAccount",
                "account:EnableRegion",
                "account:DisableRegion"
            ],
            "Resource": "*"
        }
    ]
}
```

Troubleshooting AWS Account Management identity and access

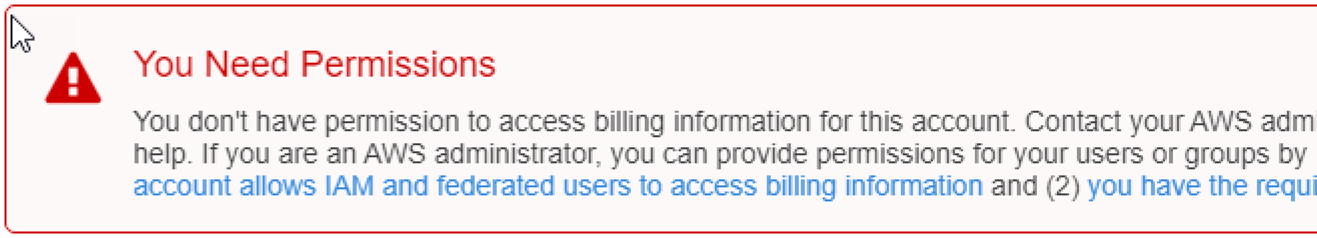
Use the following information to help you diagnose and fix common issues that you might encounter when working with Account Management and IAM.

Topics

I am not authorized to perform an action in the Accounts Settings page

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about his AWS account in the `Account Settings` page of the AWS Management Console but doesn't have the `aws-portal:ViewAccount` permissions.



You Need Permissions

You don't have permission to access billing information for this account. Contact your AWS adm
help. If you are an AWS administrator, you can provide permissions for your users or groups by
account allows IAM and federated users to access billing information and (2) you have the requi

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

I am not authorized to perform `iam:PassRole`

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Account Management.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Account Management. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

> **Important**
> Do not provide your access keys to a third party, even to help find your canonical user ID. By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing access keys in the *IAM User Guide*.

I'm an administrator and want to allow others to access my account details

To allow others to access Account Management, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in Account Management.

To get started right away, see Creating your first IAM delegated user and group in the *IAM User Guide*.

I want to allow people outside of my AWS account to access my account details

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Account Management supports these features, see How AWS Account Management works with IAM (p. 58).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

AWS managed policies for AWS Account Management

AWS Account Management currently provides two AWS managed policies that are available for your use:

- AWS managed policy: AWSAccountManagementReadOnlyAccess (p. 68)
- AWS managed policy: AWSAccountManagementFullAccess (p. 68)
- Account Management updates to AWS managed policies (p. 69)

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to create IAM customer managed policies that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see AWS managed policies in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the `ViewOnlyAccess` AWS managed policy provides read-only access to many AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see AWS managed policies for job functions in the *IAM User Guide*.

AWS managed policy: AWSAccountManagementReadOnlyAccess

You can attach the `AWSAccountManagementReadOnlyAccess` policy to your IAM identities.

This policy provides read-only permissions to only view the following:

- The metadata about your AWS accounts
- The AWS Regions that are enabled or disabled for the AWS account (you can view status of Regions in your account only by using the AWS console)

It does this by granting permission to run any of the `Get*` or `List*` operations. It doesn't provide any ability to modify the account metadata or enable or disable AWS Regions for the account.

Permissions details

This policy includes the following permissions.

- `account` – Allows principals to retrieve the metadata information about AWS accounts. It also allows principals to list the AWS Regions that are enabled for the account in the AWS Management Console.

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

AWS managed policy: AWSAccountManagementFullAccess

You can attach the `AWSAccountManagementFullAccess` policy to your IAM identities.

This policy provides full administrative access to view or modify the following:

- The metadata about your AWS accounts
- The AWS Regions that are enabled or disabled for the AWS account (you can view status or enable or disable Regions for your account only by using the AWS console)

It does this by granting permission to run any `account` operations.

Permissions details

This policy includes the following permissions.

- `account` – Allows principals to view or modify the metadata information about AWS accounts. It also allows principals to list the AWS Regions that are enabled for the account and enable or disable them in the AWS Management Console.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "account:*",
            "Resource": "*"
        }
    ]
}
```

Account Management updates to AWS managed policies

View details about updates to AWS managed policies for Account Management since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Account Management Document history page.

Change	Description	Date
AWS Account Management launched with new AWS managed policies and started tracking changes	Account Management initially launched with the following AWS managed policies: • AWSAccountManagementReadOnlyAccess (p. 68) • AWSAccountManagementFullAccess (p. 68)	Sept 30, 2021

Compliance validation for AWS Account Management

Third-party auditors assess the security and compliance of AWS services that can run in your AWS account as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS services in scope by compliance program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact in the *AWS Artifact User Guide*.

Your compliance responsibility when using services in your AWS account is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – AWS Config assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in AWS Account Management

The AWS global infrastructure is built around AWS Regions and Availability Zones. Regions provide multiple physically separated and isolated Availability Zones, which are connected through low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure security in AWS Account Management

As managed services, AWS services running in your AWS account are protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access your account settings through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed using an access key ID and a secret access key that is associated with an IAM principal. You can also use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Monitoring AWS Account Management

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS Account Management and your other AWS solutions. AWS provides the following monitoring tools to watch Account Management, report when something is wrong, and take automatic actions when appropriate:

- *AWS CloudTrail* captures (logs) API calls and related events made by or on behalf of your AWS account and writes the log files to an Amazon Simple Storage Service (Amazon S3) bucket that you specify. This lets you identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide.

Logging AWS Account Management API calls using AWS CloudTrail

The AWS Account Management APIs are integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service that calls an Account Management operation. CloudTrail captures all Account Management API calls as events. The calls captured include all calls to the Account Management operations. If you create a trail, you can turn on continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Account Management operations. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that called an Account Management operation, the IP address used to make the request, who made the request and when, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Account Management information in CloudTrail

CloudTrail is turned on in your AWS account when you create the account. When activity occurs with an Account Management operation, CloudTrail records that activity in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for Account Management operations, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the AWS Management Console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. You can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple Regions
- Receiving CloudTrail log files from multiple accounts

AWS CloudTrail logs all Account Management API operations found in the API Reference (p. 75) section of this guide. For example, calls to the `CreateAccount`, `DeleteAlternateContact`, and `PutAlternateContact` operations generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root user or AWS Identity and Access Management (IAM) user credentials
- Whether the request was made with temporary security credentials for an IAM role or federated user
- Whether the request was made by another AWS service

For more information, see the CloudTrail userIdentity element.

Understanding the Account Management log entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested operation, the date and time of the operation, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

Example 1: The following example shows a CloudTrail log entry for a call to the `GetAlternateContact` operation to retrieve the current `OPERATIONS` alternate contact for an account. The values returned by the operation aren't included in the logged information.

Example Example 1

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROA1234567890EXAMPLE:AccountAPITests",
    "arn":"arn:aws:sts::123456789012:assumed-role/ServiceTestRole/AccountAPITests",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROA1234567890EXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/ServiceTestRole",
        "accountId": "123456789012",
        "userName": "ServiceTestRole"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2021-04-30T19:25:53Z"
      }
    }
  },
  "eventTime": "2021-04-30T19:26:15Z",
  "eventSource": "account.amazonaws.com",
  "eventName": "GetAlternateContact",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "10.24.34.250",
  "userAgent": "Mozilla/5.0",
  "requestParameters": {
    "alternateContactType": "SECURITY"
  },
  "responseElements": null,
  "requestID": "1a2b3c4d-5e6f-1234-abcd-111111111111",
  "eventID": "1a2b3c4d-5e6f-1234-abcd-222222222222",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "123456789012"
}
```

Example 2: The following example shows a CloudTrail log entry for a call to the `PutAlternateContact` operation to add a new `BILLING` alternate contact to an account.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROA1234567890EXAMPLE:AccountAPITests",
    "arn": "arn:aws:sts::123456789012:assumed-role/ServiceTestRole/AccountAPITests",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROA1234567890EXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/ServiceTestRole",
        "accountId": "123456789012",
        "userName": "ServiceTestRole"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2021-04-30T18:33:00Z"
      }
    }
  },
  "eventTime": "2021-04-30T18:33:08Z",
  "eventSource": "account.amazonaws.com",
  "eventName": "PutAlternateContact",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "10.24.34.250",
  "userAgent": "Mozilla/5.0",
  "requestParameters": {
    "name": "*Alejandro Rosalez*",
    "emailAddress": "alrosalez@example.com",
    "title": "CFO",
    "alternateContactType": "BILLING"
  },
  "responseElements": null,
  "requestID": "1a2b3c4d-5e6f-1234-abcd-333333333333",
  "eventID": "1a2b3c4d-5e6f-1234-abcd-444444444444",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "123456789012"
}
```

Example 3: The following example shows a CloudTrail log entry for a call to the `DeleteAlternateContact` operation to delete the current `OPERATIONS` alternate contact.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROA1234567890EXAMPLE:AccountAPITests",
    "arn":"arn:aws:sts::123456789012:assumed-role/ServiceTestRole/AccountAPITests",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROA1234567890EXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/ServiceTestRole",
        "accountId": "123456789012",
        "userName": "ServiceTestRole"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2021-04-30T18:33:00Z"
      }
    }
  },
  "eventTime": "2021-04-30T18:33:16Z",
  "eventSource": "account.amazonaws.com",
  "eventName": "DeleteAlternateContact",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "10.24.34.250",
  "userAgent": "Mozilla/5.0",
  "requestParameters": {
    "alternateContactType": "OPERATIONS"
  },
  "responseElements": null,
  "requestID": "1a2b3c4d-5e6f-1234-abcd-555555555555",
  "eventID": "1a2b3c4d-5e6f-1234-abcd-666666666666",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "123456789012"
}
```

API Reference

The API operations in the Account Management (`account`) namespace enable you to modify your AWS account.

Every AWS account supports metadata with information about the account, including information about up to three alternate contacts associated with the account. These are in addition to the email address associated with the root user of the account. You can specify only one of each of the following contact types associated with an account.

- Billing contact
- Operations contact
- Security contact

By default, the API operations discussed in this guide apply directly to the account that calls the operation. The identity in the account that is calling the operation is typically an IAM role or IAM user and must have permission applied by an IAM policy to call the API operation. Alternatively, you can call these API operations from an identity in an AWS Organizations management account and specify the account ID number for any AWS account that is a member of the organization.

API version

This version of the Accounts API Reference documents the Account Management API version 2021-02-01.

> **Note**
> As an alternative to using the API directly, you can use one of the AWS SDKs, which consist of libraries and sample code for various programming languages and platforms (Java, Ruby, .NET, iOS, Android, and more). The SDKs provide a convenient way to create programmatic access to AWS Organizations. For example, the SDKs take care of cryptographically signing requests, managing errors, and retrying requests automatically. For more information about the AWS SDKs, including how to download and install them, see Tools for Amazon Web Services.

We recommend that you use the AWS SDKs to make programmatic API calls to the Account Management service. However, you also can use the Account Management Query API to make direct calls to the Account Management web service. To learn more about the Account Management Query API, see Making HTTP Query Requests in the Account Management User Guide. Organizations supports GET and POST requests for all actions. That is, the API does not require you to use GET for some actions and POST for others. However, GET requests are subject to the limitation size of a URL. Therefore, for operations that require larger sizes, use a POST request.

Signing requests

When you send HTTP requests to AWS, you must sign the requests so that AWS can identify who sent them. You sign requests with your AWS access key, which consists of an access key ID and a secret access key. We strongly recommend that you do not create an access key for your root account. Anyone who has the access key for your root account has unrestricted access to all the resources in your account. Instead, create an access key for an IAM user account that has administrative privileges. As another option, use AWS Security Token Service to generate temporary security credentials, and use those credentials to sign requests.

To sign requests, we recommend that you use Signature Version 4. If you have an existing application that uses Signature Version 2, you do not have to update it to use Signature Version 4. However, some operations now require Signature Version 4. The documentation for operations that require version 4 indicate this requirement.

When you use the AWS Command Line Interface (AWS CLI) or one of the AWS SDKs to make requests to AWS, these tools automatically sign the requests for you with the access key that you specify when you configure the tools.

Support and feedback for Account Management

We welcome your feedback. Send your comments to feedback-awsaccounts@amazon.com or post your feedback and questions in the Account Management support forum. For more information about the AWS support forums, see Forums Help.

How examples are presented

The JSON returned by the Account Management as response to your requests is returned as a single long string without line breaks or formatting whitespace. Both line breaks and whitespace are shown in the examples in this guide to improve readability. When example input parameters also would result in long strings that would extend beyond the screen, we insert line breaks to enhance readability. You should always submit the input as a single JSON text string.

Recording API Requests

Account Management supports CloudTrail, a service that records AWS API calls for your AWS account and delivers log files to an Amazon S3 bucket. By using information collected by CloudTrail, you can determine which requests were successfully made to Account Management, who made the request, when it was made, and so on. For more about Account Management and its support for CloudTrail, see Logging AWS Account Management API calls using AWS CloudTrail (p. 71). To learn more about CloudTrail, including how to turn it on and find your log files, see the AWS CloudTrail User Guide.

Actions

The following actions are supported:

- DeleteAlternateContact (p. 77)
- GetAlternateContact (p. 80)
- GetContactInformation (p. 84)
- PutAlternateContact (p. 87)
- PutContactInformation (p. 91)

DeleteAlternateContact

Deletes the specified alternate contact from an AWS account.

For complete details about how to use the alternate contact operations, see Access or updating the alternate contacts.

> **Note**
> Before you can update the alternate contact information for an AWS account that is managed by AWS Organizations, you must first enable integration between AWS Account Management and Organizations. For more information, see Enabling trusted access for AWS Account Management.

Request Syntax

```
POST /deleteAlternateContact HTTP/1.1
Content-type: application/json

{
   "AccountId": "string",
   "AlternateContactType": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

AccountId (p. 77)

Specifies the 12 digit account ID number of the AWS account that you want to access or modify with this operation.

If you do not specify this parameter, it defaults to the AWS account of the identity used to call the operation.

To use this parameter, the caller must be an identity in the organization's management account or a delegated administrator account, and the specified account ID must be a member account in the same organization. The organization must have all features enabled, and the organization must have trusted access enabled for the Account Management service, and optionally a delegated admin account assigned.

> **Note**
> The management account can't specify its own `AccountId`; it must call the operation in standalone context by not including the `AccountId` parameter.

To call this operation on an account that is not a member of an organization, then don't specify this parameter, and call the operation using an identity belonging to the account whose contacts you wish to retrieve or modify.

Type: String

Pattern: `^\d{12}$`

Required: No

AlternateContactType (p. 77)

Specifies which of the alternate contacts to delete.

Type: String

Valid Values: `BILLING | OPERATIONS | SECURITY`

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 99).

AccessDeniedException

The operation failed because the calling identity doesn't have the minimum required permissions.

HTTP Status Code: 403

InternalServerException

The operation failed because of an error internal to AWS. Try your operation again later.

HTTP Status Code: 500

ResourceNotFoundException

The operation failed because it specified a resource that can't be found.

HTTP Status Code: 404

TooManyRequestsException

The operation failed because it was called too frequently and exceeded a throttle limit.

HTTP Status Code: 429

ValidationException

The operation failed because one of the input parameters was invalid.

HTTP Status Code: 400

Examples

Example 1

The following example deletes the security alternate contact for the account whose credentials are used to call the operation.

Sample Request

```
POST / HTTP/1.1
X-Amz-Target: AWSAccountV20210201.DeleteAlternateContact

{ "AlternateContactType": "SECURITY" }
```

Sample Response

```
HTTP/1.1 200 OK
Content-Type: application/json
```

Example 2

The following example deletes the billing alternate contact for the specified member account in an organization. You must use credentials from either the organization's management account or from the Account Management service's delegated admin account.

Sample Request

```
POST / HTTP/1.1
X-Amz-Target: AWSAccountV20210201.DeleteAlternateContact

{ "AccountId": "123456789012", "AlternateContactType": "BILLING" }
```

Sample Response

```
HTTP/1.1 200 OK
Content-Type: application/json
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetAlternateContact

Retrieves the specified alternate contact attached to an AWS account.

For complete details about how to use the alternate contact operations, see Access or updating the alternate contacts.

> **Note**
> Before you can update the alternate contact information for an AWS account that is managed by AWS Organizations, you must first enable integration between AWS Account Management and Organizations. For more information, see Enabling trusted access for AWS Account Management.

Request Syntax

```
POST /getAlternateContact HTTP/1.1
Content-type: application/json

{
   "AccountId": "string",
   "AlternateContactType": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

AccountId (p. 80)

Specifies the 12 digit account ID number of the AWS account that you want to access or modify with this operation.

If you do not specify this parameter, it defaults to the AWS account of the identity used to call the operation.

To use this parameter, the caller must be an identity in the organization's management account or a delegated administrator account, and the specified account ID must be a member account in the same organization. The organization must have all features enabled, and the organization must have trusted access enabled for the Account Management service, and optionally a delegated admin account assigned.

> **Note**
> The management account can't specify its own `AccountId`; it must call the operation in standalone context by not including the `AccountId` parameter.

To call this operation on an account that is not a member of an organization, then don't specify this parameter, and call the operation using an identity belonging to the account whose contacts you wish to retrieve or modify.

Type: String

Pattern: `^\d{12}$`

Required: No

AlternateContactType (p. 80)

Specifies which alternate contact you want to retrieve.

Type: String

Valid Values: `BILLING | OPERATIONS | SECURITY`

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "AlternateContact": {
      "AlternateContactType": "string",
      "EmailAddress": "string",
      "Name": "string",
      "PhoneNumber": "string",
      "Title": "string"
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

AlternateContact (p. 81)

A structure that contains the details for the specified alternate contact.

Type: AlternateContact (p. 94) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 99).

AccessDeniedException

The operation failed because the calling identity doesn't have the minimum required permissions.

HTTP Status Code: 403

InternalServerException

The operation failed because of an error internal to AWS. Try your operation again later.

HTTP Status Code: 500

ResourceNotFoundException

The operation failed because it specified a resource that can't be found.

HTTP Status Code: 404

TooManyRequestsException

The operation failed because it was called too frequently and exceeded a throttle limit.

HTTP Status Code: 429

ValidationException

The operation failed because one of the input parameters was invalid.

HTTP Status Code: 400

Examples

Example 1

The following example retrieves the security alternate contact for the account whose credentials are used to call the operation.

Sample Request

```
POST / HTTP/1.1
X-Amz-Target: AWSAccountV20210201.GetAlternateContact

{ "AlternateContactType": "SECURITY" }
```

Sample Response

```
HTTP/1.1 200 OK
Content-Type: application/json{
   "AlternateContact": {
        "Name": "Anika",
        "Title": "COO",
        "EmailAddress": "anika@example.com",
        "PhoneNumber": "206-555-0198"
        "AlternateContactType": "Security"
    }
}
```

Example 2

The following example retrieves the operations alternate contact for the specified member account in an organization. You must use credentials from either the organization's management account or from the Account Management service's delegated admin account.

Sample Request

```
POST / HTTP/1.1
X-Amz-Target: AWSAccountV20210201.GetAlternateContact

{ "AccountId": "123456789012", "AlternateContactType": "Operations" }
```

Sample Response

```
HTTP/1.1 200 OK
Content-Type: application/json{
   "AlternateContact": {
        "Name": "Anika",
        "Title": "COO",
        "EmailAddress": "anika@example.com",
        "PhoneNumber": "206-555-0198"
        "AlternateContactType": "Operations"
    }
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetContactInformation

Retrieves the primary contact information of an AWS account.

For complete details about how to use the primary contact operations, see Update the primary and alternate contact information.

Request Syntax

```
POST /getContactInformation HTTP/1.1
Content-type: application/json

{
   "AccountId": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

AccountId (p. 84)

Specifies the 12-digit account ID number of the AWS account that you want to access or modify with this operation. If you don't specify this parameter, it defaults to the AWS account of the identity used to call the operation. To use this parameter, the caller must be an identity in the organization's management account or a delegated administrator account. The specified account ID must also be a member account in the same organization. The organization must have all features enabled, and the organization must have trusted access enabled for the Account Management service, and optionally a delegated admin account assigned.

> **Note**
> The management account can't specify its own `AccountId`. It must call the operation in standalone context by not including the `AccountId` parameter.

To call this operation on an account that is not a member of an organization, don't specify this parameter. Instead, call the operation using an identity belonging to the account whose contacts you wish to retrieve or modify.

Type: String

Pattern: `^\d{12}$`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "ContactInformation": {
      "AddressLine1": "string",
      "AddressLine2": "string",
      "AddressLine3": "string",
      "City": "string",
      "CompanyName": "string",
      "CountryCode": "string",
      "DistrictOrCounty": "string",
      "FullName": "string",
      "PhoneNumber": "string",
      "PostalCode": "string",
      "StateOrRegion": "string",
      "WebsiteUrl": "string"
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ContactInformation (p. 84)

Contains the details of the primary contact information associated with an AWS account.

Type: ContactInformation (p. 96) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 99).

AccessDeniedException

The operation failed because the calling identity doesn't have the minimum required permissions.

HTTP Status Code: 403

InternalServerException

The operation failed because of an error internal to AWS. Try your operation again later.

HTTP Status Code: 500

ResourceNotFoundException

The operation failed because it specified a resource that can't be found.

HTTP Status Code: 404

TooManyRequestsException

The operation failed because it was called too frequently and exceeded a throttle limit.

HTTP Status Code: 429

ValidationException

The operation failed because one of the input parameters was invalid.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

PutAlternateContact

Modifies the specified alternate contact attached to an AWS account.

For complete details about how to use the alternate contact operations, see Access or updating the alternate contacts.

> **Note**
> Before you can update the alternate contact information for an AWS account that is managed by AWS Organizations, you must first enable integration between AWS Account Management and Organizations. For more information, see Enabling trusted access for AWS Account Management.

Request Syntax

```
POST /putAlternateContact HTTP/1.1
Content-type: application/json

{
   "AccountId": "string",
   "AlternateContactType": "string",
   "EmailAddress": "string",
   "Name": "string",
   "PhoneNumber": "string",
   "Title": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

AccountId (p. 87)

Specifies the 12 digit account ID number of the AWS account that you want to access or modify with this operation.

If you do not specify this parameter, it defaults to the AWS account of the identity used to call the operation.

To use this parameter, the caller must be an identity in the organization's management account or a delegated administrator account, and the specified account ID must be a member account in the same organization. The organization must have all features enabled, and the organization must have trusted access enabled for the Account Management service, and optionally a delegated admin account assigned.

> **Note**
> The management account can't specify its own `AccountId`; it must call the operation in standalone context by not including the `AccountId` parameter.

To call this operation on an account that is not a member of an organization, then don't specify this parameter, and call the operation using an identity belonging to the account whose contacts you wish to retrieve or modify.

Type: String

Pattern: `^\d{12}$`

Required: No

AlternateContactType (p. 87)

Specifies which alternate contact you want to create or update.

Type: String

Valid Values: `BILLING | OPERATIONS | SECURITY`

Required: Yes

EmailAddress (p. 87)

Specifies an email address for the alternate contact.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `^[\s]*[\w+=.#!&-]+@[\w.-]+\.[\w]+[\s]*$`

Required: Yes

Name (p. 87)

Specifies a name for the alternate contact.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

PhoneNumber (p. 87)

Specifies a phone number for the alternate contact.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 25.

Pattern: `^[\s0-9()+-]+$`

Required: Yes

Title (p. 87)

Specifies a title for the alternate contact.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 50.

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 99).

AccessDeniedException

The operation failed because the calling identity doesn't have the minimum required permissions.

HTTP Status Code: 403

InternalServerException

The operation failed because of an error internal to AWS. Try your operation again later.

HTTP Status Code: 500

TooManyRequestsException

The operation failed because it was called too frequently and exceeded a throttle limit.

HTTP Status Code: 429

ValidationException

The operation failed because one of the input parameters was invalid.

HTTP Status Code: 400

Examples

Example 1

The following example sets the billing alternate contact for the account whose credentials are used to call the operation.

Sample Request

```
POST / HTTP/1.1
X-Amz-Target: AWSAccountV20210201.PutAlternateContact

{
    "AlternateContactType": "Billing",
    "Name": "Carlos Salazar",
    "Title": "CFO",
    "EmailAddress": "carlos@example.com",
    "PhoneNumber": "206-555-0199"
}
```

Sample Response

```
HTTP/1.1 200 OK
Content-Type: application/json
```

Example 2

The following example sets or overwrites the billing alternate contact for the specified member account in an organization. You must use credentials from either the organization's management account or from the Account Management service's delegated admin account.

Sample Request

```
POST / HTTP/1.1
X-Amz-Target: AWSAccountV20210201.PutAlternateContact

{
    "AccountId": "123456789012",
    "AlternateContactType": "Billing",
    "Name": "Carlos Salazar",
    "Title": "CFO",
    "EmailAddress": "carlos@example.com",
    "PhoneNumber": "206-555-0199"
}
```

Sample Response

```
HTTP/1.1 200 OK
Content-Type: application/json
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

PutContactInformation

Updates the primary contact information of an AWS account.

For complete details about how to use the primary contact operations, see Update the primary and alternate contact information.

Request Syntax

```
POST /putContactInformation HTTP/1.1
Content-type: application/json

{
   "AccountId": "string",
   "ContactInformation": {
      "AddressLine1": "string",
      "AddressLine2": "string",
      "AddressLine3": "string",
      "City": "string",
      "CompanyName": "string",
      "CountryCode": "string",
      "DistrictOrCounty": "string",
      "FullName": "string",
      "PhoneNumber": "string",
      "PostalCode": "string",
      "StateOrRegion": "string",
      "WebsiteUrl": "string"
   }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

AccountId (p. 91)

Specifies the 12-digit account ID number of the AWS account that you want to access or modify with this operation. If you don't specify this parameter, it defaults to the AWS account of the identity used to call the operation. To use this parameter, the caller must be an identity in the organization's management account or a delegated administrator account. The specified account ID must also be a member account in the same organization. The organization must have all features enabled, and the organization must have trusted access enabled for the Account Management service, and optionally a delegated admin account assigned.

> **Note**
> The management account can't specify its own `AccountId`. It must call the operation in standalone context by not including the `AccountId` parameter.

To call this operation on an account that is not a member of an organization, don't specify this parameter. Instead, call the operation using an identity belonging to the account whose contacts you wish to retrieve or modify.

Type: String

Pattern: `^\d{12}$`

Required: No

ContactInformation (p. 91)

Contains the details of the primary contact information associated with an AWS account.

Type: ContactInformation (p. 96) object

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 99).

AccessDeniedException

The operation failed because the calling identity doesn't have the minimum required permissions.

HTTP Status Code: 403

InternalServerException

The operation failed because of an error internal to AWS. Try your operation again later.

HTTP Status Code: 500

TooManyRequestsException

The operation failed because it was called too frequently and exceeded a throttle limit.

HTTP Status Code: 429

ValidationException

The operation failed because one of the input parameters was invalid.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3

- AWS SDK for Python
- AWS SDK for Ruby V3

Related actions in other AWS services

The following operations are related to AWS Account Management but are part of the AWS Organizations namespace:

- CreateAccount (p. 93)
- CreateGovCloudAccount (p. 93)
- DescribeAccount (p. 93)

CreateAccount

The `CreateAccount` API operation is available to use only in the context of an organization that is managed by the AWS Organizations service. The API operation is defined in that service's namespace.

For more information, see CreateAccount in the *AWS Organizations API Reference*.

CreateGovCloudAccount

The `CreateGovCloudAccount` API operation is available to use only in the context of an organization that is managed by the AWS Organizations service. The API operation is defined in that service's namespace.

For more information, see CreateGovCloudAccount in the *AWS Organizations API Reference*.

DescribeAccount

The `DescribeAccount` API operation is available to use only in the context of an organization that is managed by the AWS Organizations service. The API operation is defined in that service's namespace.

For more information, see DescribeAccount in the *AWS Organizations API Reference*.

Data Types

The following data types are supported:

- AlternateContact (p. 94)
- ContactInformation (p. 96)

AlternateContact

A structure that contains the details of an alternate contact associated with an AWS account

Contents

AlternateContactType

The type of alternate contact.

Type: String

Valid Values: `BILLING | OPERATIONS | SECURITY`

Required: No

EmailAddress

The email address associated with this alternate contact.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `^[\s]*[\w+=.#!&-]+@[\w.-]+\.[\w]+[\s]*$`

Required: No

Name

The name associated with this alternate contact.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Required: No

PhoneNumber

The phone number associated with this alternate contact.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 25.

Pattern: `^[\s0-9()+-]+$`

Required: No

Title

The title associated with this alternate contact.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 50.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ContactInformation

Contains the details of the primary contact information associated with an AWS account.

Contents

AddressLine1

The first line of the primary contact address.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 60.

Required: Yes

AddressLine2

The second line of the primary contact address, if any.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 60.

Required: No

AddressLine3

The third line of the primary contact address, if any.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 60.

Required: No

City

The city of the primary contact address.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 50.

Required: Yes

CompanyName

The name of the company associated with the primary contact information, if any.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 50.

Required: No

CountryCode

The ISO-3166 two-letter country code for the primary contact address.

Type: String

Length Constraints: Fixed length of 2.

Required: Yes

DistrictOrCounty

The district or county of the primary contact address, if any.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 50.

Required: No

FullName

The full name of the primary contact address.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 50.

Required: Yes

PhoneNumber

The phone number of the primary contact information. The number will be validated and, in some countries, checked for activation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 20.

Pattern: `^[+][\s0-9()-]+$`

Required: Yes

PostalCode

The postal code of the primary contact address.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 20.

Required: Yes

StateOrRegion

The state or region of the primary contact address. This field is required in selected countries.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 50.

Required: No

WebsiteUrl

The URL of the website associated with the primary contact information, if any.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Common Parameters

The following list contains the parameters that all actions use for signing Signature Version 4 requests with a query string. Any action-specific parameters are listed in the topic for that action. For more information about Signature Version 4, see Signature Version 4 Signing Process in the *Amazon Web Services General Reference*.

Action

The action to be performed.

Type: string

Required: Yes

Version

The API version that the request is written for, expressed in the format YYYY-MM-DD.

Type: string

Required: Yes

X-Amz-Algorithm

The hash algorithm that you used to create the request signature.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Valid Values: `AWS4-HMAC-SHA256`

Required: Conditional

X-Amz-Credential

The credential scope value, which is a string that includes your access key, the date, the region you are targeting, the service you are requesting, and a termination string ("aws4_request"). The value is expressed in the following format: *access_key*/*YYYYMMDD*/*region*/*service*/aws4_request.

For more information, see Task 2: Create a String to Sign for Signature Version 4 in the *Amazon Web Services General Reference*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-Date

The date that is used to create the signature. The format must be ISO 8601 basic format (YYYYMMDD'T'HHMMSS'Z'). For example, the following date time is a valid X-Amz-Date value: `20120325T120000Z`.

Condition: X-Amz-Date is optional for all requests; it can be used to override the date used for signing requests. If the Date header is specified in the ISO 8601 basic format, X-Amz-Date is not required. When X-Amz-Date is used, it always overrides the value of the Date header. For more information, see Handling Dates in Signature Version 4 in the *Amazon Web Services General Reference*.

Type: string

Required: Conditional

X-Amz-Security-Token

The temporary security token that was obtained through a call to AWS Security Token Service (AWS STS). For a list of services that support temporary security credentials from AWS Security Token Service, go to AWS Services That Work with IAM in the *IAM User Guide*.

Condition: If you're using temporary security credentials from the AWS Security Token Service, you must include the security token.

Type: string

Required: Conditional

X-Amz-Signature

Specifies the hex-encoded signature that was calculated from the string to sign and the derived signing key.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-SignedHeaders

Specifies all the HTTP headers that were included as part of the canonical request. For more information about specifying signed headers, see Task 1: Create a Canonical Request For Signature Version 4 in the *Amazon Web Services General Reference*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

Common Errors

This section lists the errors common to the API actions of all AWS services. For errors specific to an API action for this service, see the topic for that API action.

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

IncompleteSignature

The request signature does not conform to AWS standards.

HTTP Status Code: 400

InternalFailure

The request processing has failed because of an unknown error, exception or failure.

HTTP Status Code: 500

InvalidAction

The action or operation requested is invalid. Verify that the action is typed correctly.

HTTP Status Code: 400

InvalidClientTokenId

The X.509 certificate or AWS access key ID provided does not exist in our records.

HTTP Status Code: 403

InvalidParameterCombination

Parameters that must not be used together were used together.

HTTP Status Code: 400

InvalidParameterValue

An invalid or out-of-range value was supplied for the input parameter.

HTTP Status Code: 400

InvalidQueryParameter

The AWS query string is malformed or does not adhere to AWS standards.

HTTP Status Code: 400

MalformedQueryString

The query string contains a syntax error.

HTTP Status Code: 404

MissingAction

The request is missing an action or a required parameter.

HTTP Status Code: 400

MissingAuthenticationToken

The request must contain either a valid (registered) AWS access key ID or X.509 certificate.

HTTP Status Code: 403

MissingParameter

A required parameter for the specified action is not supplied.

HTTP Status Code: 400

NotAuthorized

You do not have permission to perform this action.

HTTP Status Code: 400

OptInRequired

The AWS access key ID needs a subscription for the service.

HTTP Status Code: 403

RequestExpired

The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.

HTTP Status Code: 400

ServiceUnavailable

The request has failed due to a temporary failure of the server.

HTTP Status Code: 503

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

ValidationError

The input fails to satisfy the constraints specified by an AWS service.

HTTP Status Code: 400

Calling the API by making HTTP Query requests

This section contains general information about using the Query API for AWS Account Management. For details about the API operations and errors, see the API Reference (p. 75).

Note

Instead of making direct calls to the AWS Account Management Query API, you can use one of the AWS SDKs. The AWS SDKs consist of libraries and sample code for various programming languages and platforms (Java, Ruby, .NET, iOS, Android, and more). The SDKs provide a convenient way to create programmatic access to AWS Account Management and AWS. For example, the SDKs take care of tasks such as cryptographically signing requests, managing errors, and retrying requests automatically. For information about the AWS SDKs, including how to download and install them, see Tools for Amazon Web Services.

With the Query API for AWS Account Management, you can call service actions. Query API requests are HTTPS requests that must contain an `Action` parameter to indicate the operation to be performed. AWS Account Management supports `GET` and `POST` requests for all operations. That is, the API doesn't require you to use `GET` for some actions and `POST` for others. However, `GET` requests are subject to the limitation size of a URL. Although this limit is browser dependent, a typical limit is 2,048 bytes. Therefore, for Query API requests that require larger sizes, you must use a `POST` request.

The response is an XML document. For details about the response, see the individual action pages in the API Reference (p. 75).

Topics

Endpoints

AWS Account Management has a single global API endpoint that is hosted in the US East (N. Virginia) AWS Region.

For more information about AWS endpoints and Regions for all services, see Regions and Endpoints in the *AWS General Reference*.

HTTPS required

Because the Query API can return sensitive information such as security credentials, you must use HTTPS to encrypt all API requests.

Signing AWS Account Management API requests

Requests must be signed using an access key ID and a secret access key. We strongly recommend that you don't use your AWS root account credentials for everyday work with AWS Account Management. You can use the credentials for an AWS Identity and Access Management (IAM) user or temporary credentials such as you use with an IAM role.

To sign your API requests, you must use AWS Signature Version 4. For information about using Signature Version 4, see Signature Version 4 Signing Process in the *AWS General Reference*.

For more information, see the following:

- AWS Security Credentials – Provides general information about the types of credentials that you can use to access AWS.
- IAM Best Practices – Offers suggestions for using the IAM service to help secure your AWS resources, including those in AWS Account Management.
- Temporary Credentials – Describes how to create and use temporary security credentials.

Quotas for AWS Account Management

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is AWS Region-specific.

Each AWS account has the following quotas related to Account Management.

Resource	Quota
Number of alternate contacts in an AWS account	3 - one each for `BILLING`, `SECURITY`, and `OPERATIONS`
Rate of `GetAlternateContact` API operations per account	10 per second, burst to 15 per second
Rate of `PutAlternateContact` API operations per account	5 per second, burst to 8 per second
Rate of `DeleteAlternateContact` API operations per account	1 per second, burst to 2 per second

Troubleshooting your AWS account

Use the information in the following topics to help you diagnose and fix issues with your AWS account.

Troubleshooting topics

Troubleshooting issues with AWS account creation

Use the information here to help you troubleshoot issues related to creating an AWS account.

Issues

I didn't receive the call from AWS to verify my new account

When you create an AWS account, you must provide a phone number on which you can receive either an SMS message or a voice call. You specify which method to use to verify the number.

If you don't receive the message or call, verify the following:

- You entered the correct telephone number and selected the correct country code during the sign-up process.
- If you're using a mobile phone, be sure that you have a cellular signal to receive SMS messages or calls.
- The information that you entered for your payment method is correct.

If you didn't receive an SMS or call to complete the identity verification process, AWS Support can help you to activate your AWS account manually. Use the following steps:

1. Be sure that you can be reached at the telephone number that you provided for your AWS account.
2. Open the AWS Support console, and then choose **Create case**.
   a. Choose **Account and billing support**.
   b. For **Type**, select **Account**.
   c. For **Category**, select **Activation**.
   d. In the **Case description** section, provide a date and time when you can be reached.
   e. In the **Contact options** section, select **Chat** for **Contact methods**.
   f. Choose **Submit**.

Note
You can create a case with AWS Support even if your AWS account isn't yet activated.

I get an error about "maximum number of failed attempts" error when I try to verify my AWS account by phone

AWS Support can help you to manually activate your account. Follow these steps:

1. Sign in to your AWS account using the email address and password that you specified when creating your account.
2. Open the AWS Support console , and then choose **Create case**.
3. Choose **Account and Billing Support**.
4. For **Type**, select **Account**.
5. For **Category**, select **Activation**.
6. In the **Case description** section, provide a date and time when you can be reached.
7. In the **Contact options** section, select **Chat** for **Contact methods**.
8. Choose **Submit**.

AWS Support will contact you and attempt to manually activate your AWS account.

Troubleshooting issues with the root user

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

Issues

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

If you can't complete tasks when you are signed in as the root user for the account, your account might be a member of an organization in AWS Organizations. If so, and your organizational administrator used a service control policy (SCP) to limit the permissions of your account, then all users, including the root user, are affected. 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 Resetting lost or forgotten passwords or access keys for AWS.

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, check for saved correspondence from no-reply@amazon.com 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, first try to recover access to the email using one of the following options:

- 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, which "catches 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. Expand **I cannot login to my AWS account** and choose **Request Support for AWS account Credentials**. Provide the information in the form and choose **Submit**.

Troubleshooting issues with the AWS account sign-in

Use the information here to help you troubleshoot issues related to signing in as a user in your AWS account.

> **Note**
> This topic is about signing in to an Amazon Web Services account. If you're having trouble signing in to your **Amazon.com** shopping account, see Amazon Customer Service instead.

Issues

- I need my AWS account ID or alias (p. 106)
- I forgot my IAM user name or password (p. 106)

I need my AWS account ID or alias

If you're an AWS Identity and Access Management (IAM) user and you're not signed in, you must ask your administrator for the AWS account ID or AWS account alias. You need this information, plus your IAM user name and password, to sign in to an AWS account.

I forgot my IAM user name or password

If you're an IAM user, your administrator provides your credentials. If you forget your password, you must ask your administrator to reset it.

For security purposes, AWS doesn't have access to view, provide, or change your credentials.

Troubleshooting other issues with AWS accounts

Use the information here to help you troubleshoot issues related to your AWS account.

Issues

- I need to change the credit card for my AWS account (p. 107)

- I need to report fraudulent AWS account activity (p. 107)
- I need to close my AWS account (p. 107)

I need to change the credit card for my AWS account

To change the credit card for your AWS account, you must be able to sign in. AWS has protections in place that require you to prove that you're the account owner. For instructions, see Managing your credit card payment methods in the *AWS Billing User Guide*.

I need to report fraudulent AWS account activity

If you suspect fraudulent activity using your AWS account and would like to make a report, see How do I report abuse of AWS resources.

If you're having trouble with a purchase made on Amazon.com, see Amazon Customer Service.

I need to close my AWS account

For help troubleshooting issues with closing your AWS account, see Closing your AWS account (p. 28).

Document history for the Account Management User Guide

The following table describes the documentation releases for AWS Account Management.

update-history-change	update-history-description	update-history-date
New contact information APIs	Support for new `GetContactInformation` and `PutContactInformation` APIs.	July 22, 2022
AWS Account Management now supports updating alternate contacts via the AWS Organizations console.	You can now update your organization's alternate contacts via AWS Organizations console using Account API permissions provided by updated AWS Organizations managed policies.	February 8, 2022
Initial release (p. 108)	Initial release of the AWS Account Management Reference Guide	September 30, 2021

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.