Security Information

AWS Control Catalog

AWS Control Catalog: Security Information

This documentation is a draft for private preview for regions in the AWS European Sovereign Cloud. Documentation content will continue to evolve. Published: December 30, 2025.

Table of Contents

What is Control Catalog?

Welcome to the Control Catalog security information guide. The Control Catalog is a part of AWS Control Tower, which lists controls for several AWS services. It is a consolidated catalog of AWS controls. You do not need to set up AWS Control Tower to use the Control Catalog.

With the Control Catalog, you can view controls according to common use cases, including security, cost, durability, and operations.

In this document, you can find security and compliance information that you need to know, as you use the APIs that are provided by Control Catalog.

The Control Catalog embodies a Control Ontology, which is a standard classification system for controls.

Ontology overview

AWS has developed a standard classification system to help classify, organize, and create mappings among controls. This ontology can be used to map controls to existing and new regulatory standards, including 24 frameworks, as well as regulatory standards such as PCI, HIPAA, and others. We also map to industry standards such as NIST and ISO, and to Amazon-specific frameworks, including the Well-Architected framework.

The ontology has four core aspects

- **Classification of controls by Control domain, Control objective, and Common controls.** The ontology helps organize and group related controls into three levels—
  - L1: Control domain,
  - L2: Control objective,
  - L3: Common control.

  These levels have a strict hierarchical relationship. That is, each domain has multiple control objectives, but each control objective must have a single parent domain. Each control objective has multiple common controls, but each common control has a single parent objective.
- **Mapping to regulatory standards.** The ontology has a concept called a *Standard control* (L4) that represents a specific requirement within a regulatory or industry standard. These Standard controls are mapped to Common controls that help address those specific requirements.

For example, **PCI-DSS v3.2.1. ID 4.1** *Use strong cryptography and security protocols to safeguard sensitive cardholder data during transmission over open, public networks* and **NIST 800.53.r5 ID SC-16** *Transmission of security and privacy attributes* are two Standard controls, both mapping to the *Encrypt data in transit* Common control.

- **Control implementations and control evidence.** The ontology has a concept of *Control implementations* (L6) that can represent either a specific control implementation in AWS, for example, an AWS Control Tower control, an AWS Security Hub CSPM check, an AWS Config rule, and so forth, or a non-technical implementation outside AWS, such as process guidance. A separate concept of *Control evidence* (L7) represents data sources that can be used as evidence for controls by AWS Audit Manager, third-party tools, or customers themselves. These evidence sources could be AWS sources such as AWS CloudTrail events, API call logs, and AWS Config rule evaluation results. Or, they could be external sources such as customer documentation.
- **The concept of a Core control (L5).** The Core control is a mapping layer that consolidates all control implementations (L6), corresponding evidence sources (L7), related standard controls (L4), and Common controls (L3) into a single holistic object. The Core control is more of a mapping document than a control itself. It helps answer the question of *show me all the info related to control X*. Each core control can have multiple control implementations (L6) and multiple evidence sources (L7).

In summary, the AWS control catalog ontology contains seven layers. Three are hierarchical classification layers (Control domains, Control objectives, Common controls). Another layer (Standard controls) describes the regulatory or industry standard requirements. A mapping layer (Core control) describes a control outcome for a given resource type. Two layers (Control implementations, Control evidences) describe the specific control implementations and evidence sources.

This ontology was designed by an AWS team of certified auditors, based on their experience working with hundreds of customers for compliance audits. The concepts of Control domains, Control objectives, Common controls, and Standard controls (L1-L4) are used industry-wide. They match common industry patterns and NIST recommendations. The remaining three layers (L5-L7) were designed based on existing AWS concepts, such as resource types and managed controls.

Access to Control Catalog

Control Catalog is available through the console and through the Control Catalog application programming interface (API). This API provides a programmatic way to identify and filter the

common controls and related metadata that are available to you as an AWS customer. For more information, see the Control Catalog API Reference.

Security in Control Catalog

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

Security is a shared responsibility between AWS and you. The 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 .
- **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 Control Catalog;. The following topics show you how to configure Control Catalog; to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Control Catalog; resources.

Topics

Data protection in Control Catalog

The AWS applies to data protection in AWS Control Catalog. 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. You are also responsible for 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 data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or 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 require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- 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 sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 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) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with AWS Control Catalog or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text 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.

Data encryption

AWS Control Catalog does not store any customer data.

Encryption at rest

AWS Control Catalog does not encrypt customer data. Because no customer data is persisted or retained by AWS Control Catalog, there are no specific guidelines for encryption at rest.

Encryption in transit

AWS Control Catalog does not encrypt customer data. Because no sensitive data is exchanged or persisted by AWS Control Catalog, there are no specific guidelines for encryption in transit.

Key management

Encryption key management does not apply to AWS Control Catalog.

Inter-network traffic privacy

Inter-network traffic privacy does not apply to AWS Control Catalog.

Identity and access management for Control Catalog

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 AWS Control Catalog 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 based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting Control Catalog identity and access)
- **Service administrator** - determine user access and submit permission requests (see How Control Catalog works with IAM)
- **IAM administrator** - write policies to manage access (see Identity-based policy examples for Control Catalog)

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM 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 defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose 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 include 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. You must specify a principal in a resource-based policy.

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.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. 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 Control Catalog works with IAM

Before you use IAM to manage access to AWS Control Catalog, learn what IAM features are available to use with AWS Control Catalog.

IAM features you can use with Control Catalog

IAM feature	AWS Control Catalog support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys	Yes
ACLs	No
ABAC (tags in policies)	No
Temporary credentials	Yes
Principal permissions	No
Service roles	No
Service-linked roles	No

To get a high-level view of how AWS Control Catalog 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 AWS Control Catalog

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 Define custom IAM permissions with customer managed 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. 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 AWS Control Catalog

To view examples of AWS Control Catalog identity-based policies, see Identity-based policy examples for Control Catalog.

Resource-based policies within AWS Control Catalog

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. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for AWS Control Catalog

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

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

Policy actions in AWS Control Catalog use the following prefix before the action:

```
controlcatalog
```

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

```
"Action": [
      "controlcatalog:ListCommonControls",
      "controlcatalog:ListDomains"
         ]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `List`, include the following action.

```
"Action": "controlcatalog:List*"
```

To view examples of AWS Control Catalog identity-based policies, see Identity-based policy examples for Control Catalog.

Policy resources for AWS Control Catalog

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. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

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

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

An AWS Control Catalog domain has the following Amazon Resource Name (ARN) format:

```
arn:${Partition}:controlcatalog:::domain/${domainId}
```

An AWS Control Catalog objective has the following ARN format:

```
arn:${Partition}:controlcatalog:::objective/${objectiveId}
```

An AWS Control Catalog common control has the following ARN format:

```
arn:${Partition}:controlcatalog:::commonControl/${commonControlId}
```

For more information about the format of ARNs, see Amazon Resource Names (ARNs).

For example, to specify the `i-1234567890abcdef0` domain in your statement, use the following ARN.

```
"Resource": "arn:aws:controlcatalog:::domain/i-1234567890abcdef0"
```

To specify all instances that belong to a specific account, use the wildcard (*).

```
"Resource": "arn:aws:controlcatalog:::domain/*"
```

Some AWS Control Catalog actions, such as those for creating resources, cannot be performed on a specific resource. In those cases, you must use the wildcard (*).

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

Some AWS Control Catalog API actions support multiple resources. For example, `ListCommonControls` accesses an common control, an objective, and a domain, so a principal must have permissions to access each of these resources. To specify multiple resources in a single statement, separate the ARNs with commas.

```
"Resource": [
      "commonControl",
      "objective",
      "domain"
```

To view examples of AWS Control Catalog identity-based policies, see Identity-based policy examples for Control Catalog.

Policy condition keys for AWS Control Catalog

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 specifies when statements execute based on defined criteria. 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. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

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

To view examples of AWS Control Catalog identity-based policies, see Identity-based policy examples for Control Catalog.

ACLs in AWS Control Catalog

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.

ABAC with AWS Control Catalog

Supports ABAC (tags in policies): No

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes called tags. You can attach tags to IAM entities and AWS resources, then design ABAC policies to allow operations when the principal's tag matches the tag on the resource.

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.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization 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 AWS Control Catalog

Supports temporary credentials: Yes

Temporary credentials provide short-term access to AWS resources and are automatically created when you use federation or switch roles. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM and AWS services that work with IAM in the *IAM User Guide*.

Cross-service principal permissions for AWS Control Catalog

Supports forward access sessions (FAS): No

Forward access sessions (FAS) use the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. For policy details when making FAS requests, see Forward access sessions.

Service roles for AWS Control Catalog

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 Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break AWS Control Catalog functionality. Edit service roles only when AWS Control Catalog provides guidance to do so.

Service-linked roles for AWS Control Catalog

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 AWS 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 Control Catalog

By default, users and roles don't have permission to create or modify AWS Control Catalog resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

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

For details about actions and resource types defined by AWS Control Catalog, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for AWS Control Catalog in the *Service Authorization Reference*.

Topics

- Policy best practices
- Allow users to view their own permissions
- Allow users to view resources from AWS Control Catalog

Policy best practices

Identity-based policies determine whether someone can create, access, or delete AWS Control Catalog 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 with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more

information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.

- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
```

```
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Allow users to view resources from AWS Control Catalog

The following policy grants permissions to list domains, objectives, and common controls from AWS Control Catalog.

JSON

```
{
"Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ManageControlCatalogAccess",
            "Effect": "Allow",
            "Action": [
                "controlcatalog:ListDomains",
                "controlcatalog:ListObjectives",
                "controlcatalog:ListCommonControls"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

Troubleshooting Control Catalog identity and access

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

Topics

- I am not authorized to perform an action in Control Catalog
- I am not authorized to perform iam:PassRole
- I want to give people outside of my AWS account access to my Control Catalog resources

I am not authorized to perform an action in Control Catalog

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `controlcatalog:`*`GetWidget`* permissions.

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

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `controlcatalog:`*`GetWidget`* action.

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

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 AWS Control Catalog.

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 AWS Control Catalog. 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 give people outside of my AWS account access to my Control Catalog resources

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 AWS Control Catalog supports these features, see How Control Catalog works with IAM.
- 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 Cross account resource access in IAM in the *IAM User Guide*.

Compliance validation for Control Catalog

To learn whether an AWS service is within the scope of specific compliance programs, see and choose the compliance program that you are interested in. For general information, see .

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. For more information about your compliance responsibility when using AWS services, see AWS Security Documentation.

Resilience in Control Catalog

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with 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 Control Catalog

As a managed service, Control Catalog is 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 Control Catalog 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 DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

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

Configuration and vulnerability analysis in Control Catalog

Configuration and IT controls are a shared responsibility between AWS and you, our customer. For more information, see the AWS shared responsibility model.

Monitoring AWS Control Catalog

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

- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can 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 Control Catalog API calls using AWS CloudTrail

As part of AWS Control Tower Control Catalog is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service.. CloudTrail captures all API calls for Control Catalog as events. The calls captured include calls directly from the AWS Control Tower console such as to enable or disable a control, and code calls to the Control Catalog API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events that pertain to the controls in Control Catalog. 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 was made to Control Catalog (by means of AWS Control Tower), the IP address from which the request was made, who made the request, when it was made, and additional details.

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

Control Catalog information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Control Catalog, that activity is recorded 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 Control Catalog, 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 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. Additionally, 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 and Receiving CloudTrail log files from multiple accounts

All Control Catalog actions are logged by CloudTrail and are documented in the Control Catalog API Reference.. For example, calls to the `ListCommonControls`, `ListObjectives`, and `ListDomains` actions 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 or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

Understanding Control Catalog log file 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 action, the date and time of the action, 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.

The following example shows a CloudTrail log entry that demonstrates the `ListDomains` action.

```
{
      eventVersion:"1.05",
      userIdentity:{
```

```
        type:"IAMUser",
        principalId:"principalId",
        arn:"arn:aws:iam::accountId:user/userName",
        accountId:"111122223333",
        accessKeyId:"accessKeyId",
        userName:"userName",
        sessionContext:{
          sessionIssuer:{
          },
          webIdFederationData:{
          },
          attributes:{
            mfaAuthenticated:"false",
            creationDate:"2020-11-19T07:32:06Z"
          }
        }
      },
      eventTime:"2020-11-19T07:32:36Z",
      eventSource:"controlcatalog.amazonaws.com",
      eventName:"ListDomains",
      awsRegion:"us-west-2",
      sourceIPAddress:"sourceIPAddress",
      userAgent:"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36",
      requestParameters: null,
      responseElements: null,
      requestID:"0d950f8c-5211-40db-8c37-2ed38ffcc894",
      eventID:"a782029a-959e-4549-81df-9f6596775cb0",
      readOnly:false,
      eventType:"AwsApiCall",
      recipientAccountId:"recipientAccountId"
}
```

Access Control Catalog using an interface endpoint (AWS PrivateLink)

You can use AWS PrivateLink to create a private connection between your VPC and Control Catalog. You can access AWS Control Catalog as if it were in your VPC, without the use of an internet gateway, NAT device, VPN connection, or Direct Connect connection. Instances in your VPC don't need public IP addresses to access Control Catalog.

You establish this private connection by creating an *interface endpoint*, powered by AWS PrivateLink. We create an endpoint network interface in each subnet that you enable for the interface endpoint. These are requester-managed network interfaces that serve as the entry point for traffic destined for Control Catalog.

For more information, see Access AWS services through AWS PrivateLink in the *AWS PrivateLink Guide*.

Considerations for AWS Control Catalog

Before you set up an interface endpoint for Control Catalog, review Considerations in the *AWS PrivateLink Guide*.

Control Catalog supports making calls to all of its API actions through the interface endpoint.

Create an interface endpoint for Control Catalog

You can create an interface endpoint for Control Catalog using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Create an interface endpoint in the *AWS PrivateLink Guide*.

Create an interface endpoint for Control Catalog using the following service name:

```
com.amazonaws.region.controlcatalog
```

If you enable private DNS for the interface endpoint, you can make API requests to Control Catalog using its default Regional DNS name. For example, `service-name.us-east-1.amazonaws.com`.

Create an endpoint policy for your interface endpoint

An endpoint policy is an IAM resource that you can attach to an interface endpoint. The default endpoint policy allows full access to Control Catalog through the interface endpoint. To control the access allowed to Control Catalog from your VPC, attach a custom endpoint policy to the interface endpoint.

An endpoint policy specifies the following information:

- The principals that can perform actions (AWS accounts, IAM users, and IAM roles).
- The actions that can be performed.
- The resources on which the actions can be performed.

For more information, see Control access to services using endpoint policies in the *AWS PrivateLink Guide*.

Example: VPC endpoint policy for Control Catalog actions

The following is an example of a custom endpoint policy. When you attach this policy to your interface endpoint, it grants access to the listed AWS Control Catalog actions for all principals on all resources.

```
{
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "controlcatalog:ListDomains",
            "controlcatalog:ListObjectives",
            "controlcatalog:ListCommonControls"
         ],
         "Resource":"*"
      }
   ]
}
```

Note

The `GetControl` and `ListControls` API operations require a different permission, the default full permission. For an example, see the Default endpoint policy.

Document history for the Control Catalog security information guide

The following table describes the documentation releases for Control Catalog.

Change	Description	Date
Initial release	Initial release of the Control Catalog APIs and security information guide.	April 8, 2024