aws

User Guide

IAM Roles Anywhere

IAM Roles Anywhere: User Guide

Table of Contents

What is IAM Roles Anywhere? 1
IAM Roles Anywhere concepts 1
Account trust boundary 2
Multi-account setups 2
Accessing IAM Roles Anywhere 2
Public key infrastructure 4
Certificate authorities 4
Certificates 4
Key management 4
Getting started 5
Step 1: Establish trust 5
Step 2: Configure roles 6
Next steps 8
Get temporary security credentials 9
Credential Helper on GitHub 11
Advanced Features 11
OS Certificate Store Integration 11
PKCS#11 Integration 13
TPM Integration 13
Diagnostic Commands 15
Available commands 16
Comparison of credential-helper commands 16
credential-process command 17
Synopsis 17
Options 18
Output 20
Examples 21
serve command 22
Synopsis 22
Options 23
Behavior 23
Examples 24
update command 25
Synopsis 25

Options ... 25
Behavior ... 26
Examples ... 27
Credential Helper Changelog ... 28
CredentialHelper version 1.7.0 ... 28
CredentialHelper version 1.6.0 ... 28
CredentialHelper version 1.5.0 ... 28
CredentialHelper version 1.4.0 ... 28
CredentialHelper version 1.3.0 ... 28
CredentialHelper version 1.2.1 ... 28
CredentialHelper version 1.2.0 ... 29
CredentialHelper version 1.1.1 ... 29
CredentialHelper version 1.1.0 ... 29
CredentialHelper version 1.0.6 ... 29
CredentialHelper version 1.0.5 ... 29
CredentialHelper version 1.0.4 ... 29
CredentialHelper version 1.0.3 ... 30
CredentialHelper version 1.0.2 ... 30
CredentialHelper version 1.0.1 ... 30
IAM Roles Anywhere trust model ... 31
Role trusts ... 31
Signature validation ... 32
Trust policy ... 33
Certificate attribute mapping ... 40
Default mapping behavior ... 41
Put attribute mappings ... 42
Delete attribute mappings ... 43
Attribute mapping and trust policy ... 44
Source identity rules ... 47
Revocation ... 47
IAM Roles Anywhere authentication ... 49
CreateSession API ... 50
Request Syntax ... 50
URI Request Parameters ... 50
Request Body ... 51
Response Syntax ... 52

Response Elements ... 53
Credentials Object ... 54
The relationship between CreateSession and AssumeRole ... 55
Signing process ... 56
Task 1: Create a canonical request ... 56
Task 2: Create a string to sign ... 60
Task 3: Calculate the signature ... 61
Task 4: Add the signature to the HTTP request ... 61
Security ... 63
Workload identities ... 64
Data protection ... 64
Encryption in transit ... 65
Key management ... 4
Inter-network traffic privacy ... 66
Identity and access management ... 66
Audience ... 66
Authenticating with identities ... 67
Managing access using policies ... 70
How IAM Roles Anywhere works with IAM ... 73
Identity-based policy examples ... 79
Troubleshooting ... 82
Using service-linked roles ... 86
AWS managed policies ... 91
Resilience ... 96
VPC endpoints (AWS PrivateLink) ... 96
Considerations for IAM Roles Anywhere VPC endpoints ... 97
Creating an interface VPC endpoint for IAM Roles Anywhere ... 97
Creating a VPC endpoint policy for IAM Roles Anywhere ... 97
IPv4 and IPv6 access ... 98
What is IPv6? ... 98
Using dual-stack policies ... 99
Adding IPv6 to a policy ... 99
Verifying your client supports IPv6 ... 101
Monitoring ... 103
Customize notification settings ... 103
Notification events ... 104

Notification channels ... 104
IAM Roles Anywhere default notification settings ... 104
Notification evaluation criteria ... 105
Configuring custom notification threshold (console) ... 105
Disabling a notification setting (console) ... 106
Monitoring with CloudWatch ... 106
Monitoring events with Amazon EventBridge ... 108
Trust anchor certificate expiration event ... 108
Intermediate or end-entity certificate expiration event ... 109
Responding to an event ... 109
Monitoring IAM Roles Anywhere notifications ... 110
Affected resources for trust anchor expiry notifications ... 110
Affected resources for end-entity certificate expiry notifications ... 111
CloudTrail logs ... 111
IAM Roles Anywhere information in CloudTrail ... 112
Understanding IAM Roles Anywhere log file entries ... 113
Monitoring with IAM Roles Anywhere subjects ... 115
AWS CloudFormation resources ... 116
IAM Roles Anywhere and AWS CloudFormation templates ... 116
Learn more about AWS CloudFormation ... 116
Quotas ... 118
Throttling ... 121
Document history ... 122

What is AWS Identity and Access Management Roles Anywhere?

You can use AWS Identity and Access Management Roles Anywhere to obtain temporary security credentials in IAM for workloads such as servers, containers, and applications that run outside of AWS. Your workloads can use the same IAM policies and IAM roles that you use with AWS applications to access AWS resources. Using IAM Roles Anywhere means you don't need to manage long-term AWS credentials for workloads running outside of AWS.

To use IAM Roles Anywhere, your workloads must use X.509 certificates issued by your certificate authority (CA). You register the CA with IAM Roles Anywhere as a trust anchor to establish trust between your public-key infrastructure (PKI) and IAM Roles Anywhere. You can also use AWS Private Certificate Authority (AWS Private CA) to create a CA and then use that to establish trust with IAM Roles Anywhere. AWS Private CA is a managed private CA service for managing your CA infrastructure and your private certificates. For more information, see What is AWS Private CA.

Topics

- IAM Roles Anywhere concepts
- Accessing IAM Roles Anywhere

IAM Roles Anywhere concepts

Learn the basic terms and concepts used in IAM Roles Anywhere.

- **Trust anchors**

  You establish trust between IAM Roles Anywhere and your certificate authority (CA) by creating a *trust anchor*. A trust anchor is a reference to either AWS Private CA or an external CA certificate. Your workloads outside of AWS authenticate with the trust anchor using certificates issued by the trusted CA in exchange for temporary AWS credentials. There can be several trust anchors in one AWS account. For more information, see IAM Roles Anywhere trust model.

- **Roles**

  An IAM role is an IAM identity that you can create in your account that has specific permissions. A role is intended to be assumable by anyone who needs it. For IAM Roles Anywhere to be able to assume a role and deliver temporary AWS credentials, the role must trust the IAM Roles

Anywhere service principal. A trust anchor is tied to the IAM role via the `aws:SourceArn` condition key that uses the trust anchor's ARN as its value in the role's trust policy. For more information, see the section called “Role trusts”.

- **Profiles**

  To specify which roles IAM Roles Anywhere assumes and what your workloads can do with the temporary credentials, you create a profile. In a profile, you can define IAM session policies, which can be managed or inline, to limit the permissions created for a session. A profile can have many IAM roles, but only one session policy. Any session returned by a CreateSession call that references the profile will have its permissions limited by the session policy.

> **Note**
>
> All IAM Roles Anywhere resources are regional and they must be created in the same account and region to be used together.

Account trust boundary

For IAM Roles Anywhere, the trust boundary is established at the account level. This means:

- Certificates issued by any trust anchor in the account can be used to assume any target role in that same account, unless you specify conditions in the role's trust policy.
- There is no automatic integration with organization-wide controls.

Multi-account setups

For information on setting up multi-account access, see: Access for an IAM user in another AWS account that you own.

Accessing IAM Roles Anywhere

AWS Management Console

You can manage your IAM Roles Anywhere resources using the browser-based console at https://console.aws.amazon.com/rolesanywhere/.

AWS Command Line Tools

You can use the AWS command line tools to issue commands at your system command line to perform IAM Roles Anywhere and other AWS tasks. This can be faster and more convenient than using the console. The command line tools can be useful if you want to build scripts to perform AWS tasks.

AWS provides the AWS Command Line Interface (AWS CLI). For information about installing and using the AWS CLI, see the AWS Command Line Interface User Guide .

AWS SDKs

The AWS software development kits (SDKs) consist of libraries and sample code for various programming languages and platforms including Java, Python, Ruby, .NET, iOS and Android, and others. The SDKs include tasks such as 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.

Public key infrastructure in IAM Roles Anywhere

AWS Identity and Access Management Roles Anywhere relies on public key infrastructure (PKI) to establish trust between your AWS account and certificate authorities that issue certificates to workloads running in your data centers. PKI involves the generation, distribution, and verification of digital certificates through public key encryption. Trust requires either uploading your CA's digital certificate as a trust anchor or referencing an existing AWS Private Certificate Authority (AWS Private CA). Once a trust anchor is created, you can use client certificates issued by that certificate to authenticate and receive temporary credentials from IAM Roles Anywhere.

Certificate authorities

Certificate authorities are entities that are trusted to issue certificates. The CA issues signed digital certificates that affirm the identity of the certificate subject and bind that identity to the public key contained in the certificate. The CA signs a certificate by hashing the contents and then encrypting the hash with the private key related to the public key in the certificate. A client application such as a web browser that needs to affirm the identity of a subject uses the public key to verify the certificate signature. It then hashes the certificate contents and compares the hashed value to the decrypted signature to determine whether they match. Certificates can have constraints on the uses of the keys associated with the certificate. For more information about trust path validation, see RFC 5280.

Certificates

Certificates, specifically X.509 certificates, bind an identity to public key, using a signature generated from a corresponding private key.

Key management

PKI uses a pair of keys to perform cryptographic operations such as encryption and generating digital signatures. One of the keys is public and is typically made available in an X.509 certificate. The other key is private and should be stored securely. Take care to ensure that your private keys are not shared.

Getting started with IAM Roles Anywhere

To use AWS Identity and Access Management Roles Anywhere for authentication to AWS from your workloads that run outside of AWS such as servers, containers, and applications, you first create a trust anchor and profile through the IAM Roles Anywhere console.

You establish trust between IAM Roles Anywhere and your certificate authority (CA) by creating a *trust anchor*. A trust anchor is a reference to either AWS Private Certificate Authority (AWS Private CA) or an external CA certificate. You can create trust anchors for each certificate authority you want to trust.

To specify which roles IAM Roles Anywhere assumes and what your workloads can do with the temporary credentials, you create a profile. In a profile, you can define permissions with IAM managed policies.

Topics

- Step 1: Establish trust
- Step 2: Configure roles
- Next steps

Step 1: Establish trust

The first step of using IAM Roles Anywhere is creating a trust anchor, which requires you to reference a certificate authority (CA) that IAM Roles Anywhere will use to validate your authentication requests. Both root and intermediate CAs can be used as trust anchors. You will have to use either a AWS Private CA resource in your account or upload your external CA certificate. Note that CA certificates that are used as trust anchors have to satisfy certain constraints. For more information, see Signature validation.

To set up a certificate authority (CA)

- Do one of the following:
  - To use a AWS Private CA resource, open the AWS Private CA console. Follow the instructions in the AWS Private CA User Guide.
  - To use an external CA, follow the instructions provided by the CA. You provide the certificate body in a later step.

> **Important**
>
> Certificates issued from public CAs cannot be used as trust anchors.

To create a trust anchor

1. Sign in to the IAM Roles Anywhere console.
2. Choose **Create a trust anchor**.
3. In **Trust anchor name**, enter a name for the trust anchor.
4. For **Certificate authority (CA) source**, do one of the following:
   - To use an AWS Private CA resource, choose **AWS Private CA**. In the **AWS Private CA** table, choose the AWS Private CA resource.
   - To use another CA, choose **External certificate bundle**. In **External certificate bundle**, paste your CA certificate body. The certificate must be in Privacy Enhanced Mail (PEM) format.
5. (Optional) Customize notification settings based on your public key infrastructure. For more information, see customize notification settings
6. (Optional) Add metadata to the trust anchor by attaching tags as key-value pairs. For more information, see Tagging AWS resources.
7. Choose **Create a trust anchor**.

Step 2: Configure roles

Before you can create an IAM Roles Anywhere profile, you need at least one IAM role that trusts the IAM Roles Anywhere service principal. Then you can create a profile that lists the roles IAM Roles Anywhere assumes. In a profile, you can also limit the permissions for a created session with IAM managed policies.

To configure a role to trust IAM Roles Anywhere

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. On the IAM roles page, choose the role you want to use.

3. On the **Trust relationships** tab, choose **Edit trust policy**.
4. Update the trust policy to include `rolesanywhere.amazonaws.com` as shown below.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "rolesanywhere.amazonaws.com"
                ]
            },
            "Action": [
              "sts:AssumeRole",
              "sts:TagSession",
              "sts:SetSourceIdentity"
            ],
            "Condition": {
              "ArnEquals": {
                "aws:SourceArn": [
                  "arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID"
                ]
              }
            }
        }
    ]
}
```

> **Important**
>
> Without a `Condition` statement present in a role trust policy, any valid certificate from the CA used as the trust anchor, or CAs subordinate to that trust anchor may be used to assume a role via IAM roles anywhere. We recommend you use `Condition` statements on both the subject and issuer attributes to ensure that only certificates that you intend to be able to assume a role can do so. For examples, see Trust policy.

For information about editing role trust policies, see Modifying a role (console) in the *IAM User Guide*.

To create a profile

1. Sign in to the IAM Roles Anywhere console.
2. Choose **Create a profile**.
3. In **Profile name**, enter a name for the profile.
4. Under **Role**, choose the role you updated the trust policy for.
5. (Optional) Configure session policies by choosing up to 10 managed policies or write an inline policy.

   Session policies limit the permissions for a created session, but do not grant permissions. For more information, see Session policies.
6. (Optional) Add metadata to the profile by attaching tags as key–value pairs. For more information, see Tagging AWS resources.
7. Choose **Create a profile**.

Next steps

You can now authenticate with IAM Roles Anywhere. Follow the instructions in Get temporary security credentials. Also consider Monitoring with IAM Roles Anywhere subjects.

Get temporary security credentials from IAM Roles Anywhere

To obtain temporary security credentials from AWS Identity and Access Management Roles Anywhere, use the credential helper tool that IAM Roles Anywhere provides. This tool is compatible with the `credential_process` feature available across the language SDKs. When used with an AWS SDK, these credentials automatically refresh before they expire, requiring no additional implementation for credential renewal. The helper manages the process of creating a signature with the certificate and calling the endpoint to obtain session credentials; it returns the credentials to the calling process in a standard JSON format.

See Temporary security credentials in IAM for more information on session credentials.

To download the credential helper tool, use the following links. Releases for Darwin and Windows on or after version 1.1.1 are signed.

Platform	Architecture	Download URL	SHA256 checksum
Linux	x86-64	https://rolesanywhere.amazonaws.com/releases/1.7.0/X86_64/Linux/aws_signing_helper	e932f029b73f97523c1dea2e78e9543d9e2753c387c4c46e77be8c1d9424db0f
Windows	x86-64	https://rolesanywhere.amazonaws.com/releases/1.7.0/X86_64/Windows/aws_signing_helper.exe	01b678ef0173dd6a9ea302c125f0ce4bfccf9ec153070f76eeb71b421d6615f5
Darwin	x86-64	https://rolesanywhere.amazonaws.com/releases/1.7.0/	7c9bcab806754572b0a980878a8

Platform	Architecture	Download URL	SHA256 checksum
		X86_64/Darwin/aws_signing_helper	84ff2fcbd4482e2118dd1cb229b3a978c7eab
Linux	Aarch64	https://rolesanywhere.amazonaws.com/releases/1.7.0/Aarch64/Linux/aws_signing_helper	800fc208b74cdb64b8c7270c41aa0c9d2f9bf5bba498539513acb30f8eb8f164
Darwin	Aarch64	https://rolesanywhere.amazonaws.com/releases/1.7.0/Aarch64/Darwin/aws_signing_helper	644ea2a4a396a2a653ae8fc10bab4eef30a6b8763aa9f2adcd6cfb96d610fd6b

Important

To get temporary credentials, you need all of the following:

- A Profile configured in AWS Identity and Access Management Roles Anywhere
- A Role ARN from IAM
- An end-entity certificate from your certificate authority
- The certificate associated private key is required in most cases, except when inferred. For example when using OS certificate stores.
- A trust anchor configured in IAM Roles Anywhere

For more information, see Getting started.

Credential Helper on GitHub

The source code for the credential helper is available on GitHub so that you can adapt the helper to your needs. We encourage you to submit pull requests for changes that you would like to have included. However, AWS doesn't provide support for running modified copies of this software.

> **Note**
>
> You can find more modes and options and for credential helper in the README.md.

Advanced Features

The credential helper supports several advanced features for working with different types of key storage and authentication mechanisms.

OS Certificate Store Integration

On Windows and macOS, the credential helper can leverage private keys and certificates stored in OS-specific secure stores:

- **Windows:** Both CNG and Cryptography APIs are supported. By default, only the user's "MY" certificate store is searched, but you can specify a different store using the `--system-store-name` option.
- **macOS:** Keychain Access is supported. The credential helper searches Keychains on the search list.

To use certificates from these stores, use the `--cert-selector` option instead of providing certificate and private key files.

macOS Keychain Integration

For securing keys through macOS Keychain, consider creating a dedicated Keychain that only the credential helper can access:

1. Create a new Keychain:

```
security create-keychain -p ${CREDENTIAL_HELPER_KEYCHAIN_PASSWORD} credential-
helper.keychain
```

2. Unlock the Keychain:

```
security unlock-keychain -p ${CREDENTIAL_HELPER_KEYCHAIN_PASSWORD} credential-
helper.keychain
```

3. Add the Keychain to the search list:

```
EXISTING_KEYCHAINS=$(security list-keychains | cut -d '"' -f2) security list-
keychains -s credential-helper.keychain $(echo ${EXISTING_KEYCHAINS} | awk -v ORS=" "
 '{print $1}')
```

4. Import your certificate and private key:

```
security import /path/to/identity.pfx -T /path/to/aws_signing_helper -P
 ${UNWRAPPING_PASSWORD} -k credential-helper.keychain
```

Note

Since the official credential helper binary is signed, but not notarized, so it isn't trusted by macOS by default. You may need to specify your Keychain password whenever the credential helper uses the private key, or choose to always allow the credential helper to use the Keychain item.

Windows CNG Integration

To use Windows CNG for key storage, import your certificate and private key into your user's "MY" certificate store:

```
certutil -user -p %UNWRAPPING_PASSWORD% -importPFX "MY" \path\to\identity.pfx
```

You can also use PowerShell cmdlets or Windows CNG/Cryptography APIs to import certificates.

PKCS#11 Integration

The credential helper supports using certificates and keys from hardware or software PKCS#11 tokens and HSMs using PKCS#11 URIs:

- Use a certificate from a PKCS#11 token:

```
--certificate 'pkcs11:manufacturer=piv_II;id=%01'
```

- Use a certificate by object name:

```
--certificate 'pkcs11:object=My%20RA%20key'
```

- Use a certificate from a file but the key from a token:

```
--certificate client-cert.pem --private-key 'pkcs11:model=SoftHSM%20v2;object=My%20RA
%20key'
```

Most Unix-based systems use p11-kit to provide consistent system-wide and per-user configuration of available PKCS#11 providers. For systems or containers that lack p11-kit, you can specify a specific PKCS#11 provider library using the `--pkcs11-lib` parameter.

If your private key object has the `CKA_ALWAYS_AUTHENTICATE` attribute set and the `CKU_CONTEXT_SPECIFIC` PIN matches the `CKU_USER` PIN, you can use the `--reuse-pin` parameter to avoid being prompted for the PIN multiple times.

> **Note**
>
> For YubiKey devices with PIV support, when a key pair and certificate exist in slots 9a or 9c, the YubiKey automatically generates an attestation certificate for the slot. To disambiguate between your certificate and the attestation certificate, use the `CKA_LABEL` (the `object` path attribute in a URI).

TPM Integration

The credential helper supports TPM wrapped keys in the `-----BEGIN TSS2 PRIVATE KEY-----` format. You can use such a file as you would any plain key file. These files are supported by both TPMv2 OpenSSL engines/providers and GnuTLS.

Important

To use these commands below you must install the tpm2-tools software package. This package provides the necessary commands for working with TPM2 devices.
This package is available on many Linux distributions through standard package managers.

To create and use a TPM key with the credential helper:

1. Create a primary key in the TPM owner hierarchy:

```
tpm2_createprimary -G rsa -g sha256 -p ${TPM_PRIMARY_KEY_PASSWORD} -c parent.ctx -P
 ${OWNER_HIERARCHY_PASSWORD}
```

2. Create a child key with the primary key as its parent:

```
tpm2_create -C parent.ctx -u child.pub -r child.priv -P ${TPM_PRIMARY_KEY_PASSWORD} -
p ${TPM_CHILD_KEY_PASSWORD}
```

3. Load the child key into the TPM and make it persistent:

```
tpm2_load -C parent.ctx -u child.pub -r child.priv -c child.ctx -P
 ${TPM_PRIMARY_KEY_PASSWORD}
CHILD_HANDLE=$(tpm2_evictcontrol -c child.ctx | cut -d ' ' -f 2 | head -n 1)
```

4. Create a CSR using the TPM key:

```
openssl req -provider tpm2 -provider default -propquery '?provider=tpm2' \
            -new -key handle:${CHILD_HANDLE} \
            -out client-csr.pem
```

5. After obtaining a certificate from your CA, use it with the credential helper:

```
./aws_signing_helper credential-process \
    --certificate /path/to/certificate/file \
    --private-key handle:${CHILD_HANDLE} \
    --role-arn ${ROLE_ARN} \
    --trust-anchor-arn ${TA_ARN} \
    --profile-arn ${PROFILE_ARN}
```

Important

When using TPM handles, it's your responsibility to clear out the persistent and temporary objects from the TPM after you no longer need them. If you load a key into the TPM that isn't password-protected, anyone with access to the machine can use that key.

Alternatively, you can use a TPM key file in the format described in the TSS2 Private Key Format. With this approach, the wrapped private key will be loaded into the TPM as a transient object and automatically flushed after use.

Note

For RSA keys used with the credential helper, ensure they have the sign attribute set. Some tools (like the IBM OpenSSL ENGINE) create RSA keys with the decrypt attribute but not the sign attribute by default. The credential helper delegates the signing operation to the TPM rather than implementing PKCS#1 v1.5 padding manually.

Diagnostic Commands

The credential helper includes diagnostic commands that can help you troubleshoot issues:

read-certificate-data

Reads and displays information about a certificate. This is useful for verifying certificate details and finding the correct certificate when multiple certificates match a selector.

```
./aws_signing_helper read-certificate-data --certificate /path/to/certificate.pem
```

Or with a certificate selector:

```
./aws_signing_helper read-certificate-data --cert-selector
 Key=x509Subject,Value=CN=Subject
```

If multiple certificates match a selector, information about each of them is printed. For PKCS#11, URIs for each matched certificate are also printed to help uniquely identify certificates.

sign-string

Signs a fixed test string to validate your private key and digest configuration:

```
./aws_signing_helper sign-string --private-key /path/to/private-key.pem
```

Or with a certificate selector:

```
./aws_signing_helper sign-string --cert-selector Key=x509Subject,Value=CN=Subject
```

Optional parameters include:

- `--digest`: Must be one of SHA256 (default), SHA384, or SHA512
- `--format`: Must be one of text (default), json, or bin

Available commands

The credential helper tool provides several commands to help you work with IAM Roles Anywhere. This section describes the available commands and their usage.

- credential-process command - Obtains temporary security credentials from IAM Roles Anywhere using your certificate and private key.
- update command - Obtains temporary security credentials from IAM Roles Anywhere and writes them directly to the AWS credentials file.
- serve command - Vends temporary security credentials from IAM Roles Anywhere through a local endpoint.

For additional commands and options, see the credential helper README on GitHub.

Comparison of credential-helper commands

Feature	credential-process	update	serve
Automatic credential refresh	No (AWS CLI calls for each command)	Yes	Yes

Feature	credential-process	update	serve
SDK integration	No (requires profile setup)	No (requires profile setup)	Yes (with environment variable)
Credentials storage	In memory	On disk	In memory
Best for	Single commands	AWS CLI usage	SDK applications

credential-process command

The credential-process command obtains temporary security credentials from IAM Roles Anywhere using your certificate and private key.

Synopsis

```
./aws_signing_helper credential-process \
    --certificate
    [--cert-selector]
    [--endpoint]
    [--region]
    [--intermediates]
    --private-key
    [--tpm-key-password]
    [--pkcs11-lib]
    [--pkcs11-pin]
    [--reuse-pin]
    [--system-store-name]
    [--reuse-latest-expiring-certificate]
    --profile-arn
    --role-arn
    [--session-duration]
    --trust-anchor-arn
    [--with-proxy]
    [--no-verify-ssl]
    [--role-session-name]
```

Options

`--certificate` (string)

Path to certificate file. Can also be a PKCS#11 URI (e.g., `pkcs11:manufacturer=piv_II;id=%01`) to use certificates from hardware or software PKCS#11 tokens/HSMs.

`--cert-selector` (string)

Specifies which certificate (and associated private key) to use from OS-specific secure stores. On Windows, this searches the user's "MY" certificate store (or another store specified with `--system-store-name`). On macOS, this searches Keychains on the search list.

The selector can be specified either through a JSON file or directly on the command line. It allows searching by certificate Subject, Issuer, and Serial Number using the keys `x509Subject`, `x509Issuer`, and `x509Serial`.

Example using a JSON file:

```
--cert-selector file://path/to/selector.json
```

Example using command line:

```
--cert-selector Key=x509Subject,Value=CN=Subject Key=x509Issuer,Value=CN=Issuer
 Key=x509Serial,Value=15D19632234BF759A32802C0DA88F9E8AFC8702D
```

`--endpoint` (string)

The IAM Roles Anywhere endpoint for the region. For a list of endpoints, see Service endpoints and quotas.

`--region` (string)

Signing region.

`--intermediates` (string)

Path to intermediate certificate bundle.

`--private-key` (string)

Specifies the private key to use for signing. This can be:

- A Path to a plaintext private key file

- A PKCS#11 URI (e.g., `pkcs11:object=My%20RA%20key`)
- A TPM wrapped key file in the `-----BEGIN TSS2 PRIVATE KEY-----` format
- A TPM handle reference (e.g., `handle:0x81000001`)

Not required when using `--cert-selector` as the private key is inferred from the OS certificate store.

`--tpm-key-password` (string)

Password for TPM-protected private keys. Required when using password-protected TPM keys.

`--pkcs11-lib` (string)

Path to a specific PKCS#11 provider library. Only needed for systems or containers that lack p11-kit, otherwise the default p11-kit-proxy provider is used.

`--pkcs11-pin` (string)

PIN for PKCS#11 token authentication. Required when using PKCS#11 tokens that require authentication.

`--reuse-pin`

Boolean flag to reuse the PKCS#11 user PIN for object-specific authentication. Useful when the private key object has the `CKA_ALWAYS_AUTHENTICATE` attribute set and the `CKU_CONTEXT_SPECIFIC` PIN matches the `CKU_USER` PIN.

`--system-store-name` (string)

Windows only: Specifies a system certificate store other than "MY" (the default) in the `CERT_SYSTEM_STORE_CURRENT_USER` context. Only used with `--cert-selector`.

`--reuse-latest-expiring-certificate`

Boolean flag to use the latest expiring certificate when multiple certificates match a certificate selector. This can be useful when you have multiple valid certificates and want to use the one with the longest remaining validity period.

`--profile-arn` (string)

Profile to pull policies, attribute mappings and other data from.

`--role-arn` (string)

Target role to assume.

`--session-duration` (int)

Duration, in seconds, for the resulting session (corresponds to the `durationSeconds` parameter in the `CreateSession` request). This is optional and can range from 900 seconds (15 minutes) up to 43200 seconds (12 hours). Please see the `Expiration` subsection of the Credentials Object section for more details on how this value is used in determining the expiration of the vended session.

`--trust-anchor-arn` (string)

Trust anchor to use for authentication.

`--with-proxy`

To use the tool with a proxy. This is a boolean flag. Note that you will have to set the `HTTPS_PROXY` environment variable with the address of the proxy server.

`--no-verify-ssl`

To disable SSL verification. This is a boolean flag.

> **Important**
>
> Note that this disables TLS host authentication, and can open the connection to man-in-the-middle attacks. This option should only be used under specific, tightly controlled scenarios, such as debugging proxy connections.

`--role-session-name`

An identifier for the role session. Please see The relationship between CreateSession and AssumeRole section for more details on how this option will affect the `CreateSession` operation.

Output

The credential helper tool will return a JSON containing the credentials. This format allows the credentials to be consumed by the external credential process supported by the `credential_process`.

```
{
```

```
    "Version":1,
    "AccessKeyId": String,
    "SecretAccessKey": String,
    "SessionToken": String,
    "Expiration": Timestamp
}
```

Examples

Example Obtain temporary security credentials

Use the following command to get temporary security credentials. Specify the Region where you want to make the request.

> **Important**
>
> Before you run this command, make sure you have your certificate and private key. To get these credentials, follow the documentation from your certificate authority.

```
$ ./aws_signing_helper credential-process \
              --certificate /path/to/certificate \
              --private-key /path/to/private-key \
              --trust-anchor-arn arn:aws:rolesanywhere:region:account:trust-
anchor/TA_ID \
              --profile-arn arn:aws:rolesanywhere:region:account:profile/PROFILE_ID \
              --role-arn arn:aws:iam::account:role/role-name-with-path
```

Example Use temporary security credentials with AWS SDKs and the AWS CLI

To use temporary security credentials with AWS SDKs and the AWS CLI, you can configure the credential helper tool as a credential process. For more information, see Sourcing credentials with an external process.

The following example shows a the `config` file that sets the helper tool as the credential process.

```
[profile developer]
    credential_process = ./aws_signing_helper credential-process --certificate /
path/to/certificate --private-key /path/to/private-key --trust-anchor-
arn arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID --profile-
```

```
arn arn:aws:rolesanywhere:region:account:profile/PROFILE_ID --role-arn
 arn:aws:iam::account:role/role-name-with-path
    region = region
```

For using this profile with any AWS SDK not mentioned below, see 'Set a named profile' from Using shared config and credentials files to globally configure AWS SDKs and tools

Example Python SDK

To specify your Roles Anywhere enabled profile for use with Python, see Boto3: Shared credentials file.

Example Java SDK

To specify your Roles Anywhere enabled profile for use with Java, see Java 2.x: Use profiles

Example C# SDK

To specify your Roles Anywhere enabled profile for use with C#, see Examples for classes SharedCredentialsFile and AWSCredentialsFactory

Example Go SDK

To specify your IAM Roles Anywhere enabled profile for use with Go, see the Specifying Profiles section in Specifying Credentials

serve command

The serve command vends temporary security credentials from IAM Roles Anywhere through a local endpoint running on localhost. This endpoint is compatible with Instance Metadata Service (IMDSv2), allowing AWS SDKs to automatically discover and use the credentials without additional configuration.

Synopsis

```
./aws_signing_helper serve \
        --certificate
        [--cert-selector]
        [--endpoint]
```

```
        [--region]
        [--intermediates]
        --private-key
        [--tpm-key-password]
        [--pkcs11-lib]
        [--pkcs11-pin]
        [--reuse-pin]
        [--system-store-name]
        [--reuse-latest-expiring-certificate]
        --profile-arn
        --role-arn
        [--session-duration]
        --trust-anchor-arn
        [--with-proxy]
        [--no-verify-ssl]
        [--role-session-name]
        [--port]
        [--hop-limit]
```

Options

The serve command accepts all the options available to the credential-process command, plus the following additional options. For details on common options like `--cert-selector`, `--tpm-key-password`, `--pkcs11-lib`, and others, see the Options section under credential-process.

`--port` (int)

The port on which the local endpoint will be exposed. Default is 9911.

`--hop-limit` (int)

The IP TTL (Time To Live) to set on response packets. Default is 64. Setting this to 1 maintains parity with EC2's IMDSv2 hop count behavior.

Behavior

When you run the serve command, the credential helper starts a local server that:

- Listens on 127.0.0.1 at the specified port (default 9911)
- Implements the same URIs and request headers as IMDSv2
- Automatically refreshes credentials five minutes before the previous set expires

- Continues running until manually terminated

> **Important**
>
> When using the serve command, be aware that any process running on the system that can reach 127.0.0.1 will be able to retrieve AWS credentials from the credential helper.

Examples

Example Start a credential server on the default port

```
$ ./aws_signing_helper serve \
                  --certificate /path/to/certificate \
                  --private-key /path/to/private-key \
                  --trust-anchor-arn arn:aws:rolesanywhere:region:account:trust-
anchor/TA_ID \
                  --profile-arn arn:aws:rolesanywhere:region:account:profile/PROFILE_ID
 \
                  --role-arn arn:aws:iam::account:role/role-name-with-path
```

Example Start a credential server on a custom port with restricted hop limit

```
$ ./aws_signing_helper serve \
                  --certificate /path/to/certificate \
                  --private-key /path/to/private-key \
                  --trust-anchor-arn arn:aws:rolesanywhere:region:account:trust-
anchor/TA_ID \
                  --profile-arn arn:aws:rolesanywhere:region:account:profile/PROFILE_ID
 \
                  --role-arn arn:aws:iam::account:role/role-name-with-path \
                  --port 8000 \
                  --hop-limit 1
```

Example Use with AWS SDKs

To use the credentials served by the credential helper with AWS SDKs, set the `AWS_EC2_METADATA_SERVICE_ENDPOINT` environment variable to point to your local endpoint:

```
$ export AWS_EC2_METADATA_SERVICE_ENDPOINT=http://127.0.0.1:9911
```

AWS SDKs will use the environment variable and the credentials from this endpoint using their credential providers without any changes to application code. The SDKs will request new AWS credentials from the credential helper's server as needed.

update command

The update command obtains temporary security credentials from IAM Roles Anywhere and writes them directly to the AWS credentials file. This allows the AWS CLI and SDKs to use the credentials without having to call the credential helper for each command.

Synopsis

```
./aws_signing_helper update \
        --certificate
        [--cert-selector]
        [--endpoint]
        [--region]
        [--intermediates]
        --private-key
        [--tpm-key-password]
        [--pkcs11-lib]
        [--pkcs11-pin]
        [--reuse-pin]
        [--system-store-name]
        [--reuse-latest-expiring-certificate]
        --profile-arn
        --role-arn
        [--session-duration]
        --trust-anchor-arn
        [--with-proxy]
        [--no-verify-ssl]
        [--role-session-name]
        [--profile]
        [--once]
```

Options

The update command accepts all the options available to the credential-process command, plus the following additional options. For details on common options like `--cert-selector`, `--tpm-key-password`, `--pkcs11-lib`, and others, see the Options section under credential-process.

`--profile` (string)

The named profile in the AWS credentials file to update. If not specified, the default profile will be updated. If the profile doesn't exist, it will be created.

`--once`

Update the credentials only once and then exit. If not specified, the command will continuously update the credentials before they expire.

Behavior

When you run the update command, the credential helper:

- Obtains temporary security credentials from IAM Roles Anywhere
- Writes the credentials directly to the AWS credentials file (~/.aws/credentials)
- Unless the `--once` flag is specified, automatically refreshes the credentials approximately five minutes before they expire
- Continues running until manually terminated (unless `--once` is specified)

Important

When using the update command, be aware that the credentials are written to a file on disk. Any user or process who can read the credentials file may be able to read and use those AWS credentials.

Note

Running the update command multiple times simultaneously, creating multiple processes, may not work as intended. There may be issues with concurrent writes to the credentials file.

Examples

Example Update the default profile continuously

```
$ ./aws_signing_helper update \
                  --certificate /path/to/certificate \
                  --private-key /path/to/private-key \
                  --trust-anchor-arn arn:aws:rolesanywhere:region:account:trust-
anchor/TA_ID \
                  --profile-arn arn:aws:rolesanywhere:region:account:profile/PROFILE_ID
 \
                  --role-arn arn:aws:iam::account:role/role-name-with-path
```

Example Update a named profile once

```
$ ./aws_signing_helper update \
                  --certificate /path/to/certificate \
                  --private-key /path/to/private-key \
                  --trust-anchor-arn arn:aws:rolesanywhere:region:account:trust-
anchor/TA_ID \
                  --profile-arn arn:aws:rolesanywhere:region:account:profile/PROFILE_ID
 \
                  --role-arn arn:aws:iam::account:role/role-name-with-path \
                  --profile developer \
                  --once
```

Example Use with AWS CLI

After running the update command, you can use the AWS CLI with the updated profile:

```
$ aws s3 ls --profile developer
```

If you updated the default profile, you don't need to specify the profile:

```
$ aws s3 ls
```

Example Use with JavaScript SDK

To specify your Roles Anywhere enabled profile for use with JavaScript, see Loading Credentials in Node.js from the Shared Credentials File

Credential Helper Changelog

CredentialHelper version 1.7.0

On Jun 2, 2025, AWS IAM Roles Anywhere released Credential Helper version 1.7.0. As part of this release, a bug was fixed that prevented adding custom roots for use in TLS, and an enhancement was introduced to make parsing logic more robust for cert selectors provided through the CLI. Dependency module versions were also upgraded to address any vulnerabilities picked up by scanners.

CredentialHelper version 1.6.0

On May 6, 2025, AWS IAM Roles Anywhere released Credential Helper version 1.6.0. As part of this release, password-protected private keys in PKCS#8 format are now supported, and a fix was introduced for a bug where the credential helper opened terminal device files when the process didn't have a controlling terminal.

CredentialHelper version 1.5.0

On April 3, 2025, AWS IAM Roles Anywhere released Credential Helper version 1.5.0. As a part of this release, an option was added to use the latest expiring certificate when multiple match a certificate selector, and some minor fixes were introduced related to PKCS#12 file support.

CredentialHelper version 1.4.0

On December 13, 2024, AWS IAM Roles Anywhere released Credential Helper version 1.4.0. As a part of this release, TPM keys are now supported on Windows systems.

CredentialHelper version 1.3.0

On November 13, 2024, AWS IAM Roles Anywhere released Credential Helper version 1.3.0. As a part of this release, TPM keys are supported for non-Windows systems, and a previously unhandled error when parsing certificate data from a file was fixed.

CredentialHelper version 1.2.1

On October 22, 2024, AWS IAM Roles Anywhere released Credential Helper version 1.2.1. As a part of this release, some security vulnerabilities were patched.

CredentialHelper version 1.2.0

On August 23, 2024, AWS IAM Roles Anywhere released Credential Helper version 1.2.0. As a part of this release, custom role session name is supported in the CreateSession request, a hop limit option is supported to limit the IP TTL on response packets for the serve command, and file updates for certificate and private key data are recognized by long-running commands and will be honored in subsequent credentialing requests.

CredentialHelper version 1.1.1

On October 12, 2023, AWS IAM Roles Anywhere released Credential Helper version 1.1.1. As a part of this release, providers in Windows are attempted to be silenced when performing signing operations. Also, Windows user certificate store names can now be explicitly provided.

CredentialHelper version 1.1.0

On September 20, 2023, AWS IAM Roles Anywhere released Credential Helper version 1.1.0. As a part of this release, PKCS#11 module integration is now supported. Also, any debug logs that previously went to stderr now go to stdout.

CredentialHelper version 1.0.6

On August 16, 2023, AWS IAM Roles Anywhere released Credential Helper version 1.0.6. As a part of this release, certificates within OS secure stores that can't be parsed are now skipped. An issue relating to mismatched regions in the ARN inputs is also now fixed.

CredentialHelper version 1.0.5

On July 25, 2023, AWS IAM Roles Anywhere released Credential Helper version 1.0.5. As a part of this release, some bugs relating to the update command were fixed. PKCS#12 containers that aren't password-protected are now supported. Lastly, MacOS Keychain Access and Windows CNG/CryptoAPI integrations are also now supported.

CredentialHelper version 1.0.4

On January 17, 2023, AWS IAM Roles Anywhere released Credential Helper version 1.0.4. As a part of this release, some bugs specific to the serve command were fixed.

CredentialHelper version 1.0.3

On December 5, 2022, AWS IAM Roles Anywhere released Credential Helper version 1.0.3. As a part of this release, the tool now supports the update and serve commands.

CredentialHelper version 1.0.2

On September 8, 2022, AWS IAM Roles Anywhere released Credential Helper version 1.0.2. As a part of this release, the tool now sets the minimum TLS version to 1.2.

CredentialHelper version 1.0.1

On July 14, 2022, AWS IAM Roles Anywhere released Credential Helper version 1.0.1. As a part of this release, the tool now has better error handling.

The IAM Roles Anywhere trust model

AWS Identity and Access Management Roles Anywhere works by bridging the trust model of IAM and Public Key Infrastructure (PKI). The model connects the role, the IAM Roles Anywhere service principal, and identities encoded in X.509 certificates, that are issued by a Certificate Authority (CA).

Role trusts

To use an IAM role with IAM Roles Anywhere, you must create a trust relationship with the IAM Roles Anywhere service principal `rolesanywhere.amazonaws.com`. To create the trust relationship, you create a *trust policy*, a JSON policy document. The policy must grant the permissions:

- `sts:AssumeRole`
- `sts:SetSourceIdentity`
- `sts:TagSession`

Sessions issued by IAM Roles Anywhere have the source identity set to the common name(s) (CN) of the subject(s) you use in end-entity certificate(s) authenticating to the target role. IAM Roles Anywhere extracts values from the subject, issuer, and Subject Alternative Name (SAN) fields of the authenticating certificate and makes them available for policy evaluation via the sourceIdentity and principal tags. The sts:SourceIdentity key is present in the request when IAM Roles Anywhere initially sets the source identity while assuming a role. You can apply more authorization constraints by using condition clauses in the policy statement. The tags will also be evaluated against policies on any resources accessed with the issued session.

Additionally, sessions issued by AWS have a Session Name property that is used as part of the assumed role ARN. IAM Roles Anywhere sets this automatically, using the hex-encoded serial number of the authenticating certificate.

To examine the contents of a certificate, use the following command:

```
$openssl x509 -text -noout -in certificate.pem
```

```
      Certificate:
```

```
    Data:
        Version: 3 (0x2)
        Serial Number:
            1f:71:c5:11:4a:11:9f:c0:cc:5a:5a:52:fb:37:20:ad
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=Amazon, OU=IAM, ST=Washington, CN=Roles Anywhere, L=Seattle
        Validity
            Not Before: Apr 20 14:13:43 2022 GMT
            Not After : Jan 10 15:13:43 2023 GMT
        Subject: CN=awsUser
        # remainder omitted...
```

In this case, the value `awsUser` becomes the source identity. The values from the `Issuer` field become principal tags in the resulting session.

For more information, see:

- IAM roles
- IAM roles terms and concepts: trust policy
- Policies and permissions in IAM
- Things to know about source identity
- Passing session tags in AWS Security Token Service

Signature validation

To authenticate a request for credentials, IAM Roles Anywhere validates the incoming signature by using the signature validation algorithm required by the key type of the certificate, for example RSA or ECDSA. After validating the signature, IAM Roles Anywhere checks that the certificate was issued by a certificate authority configured as a trust anchor in the account using algorithms defined by public key infrastructure X.509 (PKIX) standards.

End entity certificates must satisfy the following constraints to be used for authentication:

- The certificates MUST be X.509v3.
- If the `CA` field is present in the basic constraints extension, its value MUST be `false`.
- The key usage MUST include `Digital Signature`.
- The signing algorithm MUST include `SHA256` or stronger. `MD5` and `SHA1` signing algorithms are rejected.

Certificates used as trust anchors must satisfy the same requirements for signature algorithm, but with the following differences:

- The key usage MUST include `Certificate Sign`, and MAY include `CRL Sign`. Certificate Revocation Lists (CRLs) are an optional feature of IAM Roles Anywhere.
- Basic constraints MUST include `CA: true`.

Trust policy

Temporary credentials for IAM roles are issued to IAM Roles Anywhere clients via the API method `CreateSession`. For the call to be authorized, the target role of the `CreateSession` API call must have an Assume Role Policy Document to trust the IAM Roles Anywhere service principal (`rolesanywhere.amazonaws.com`).

> **Important**
>
> It is also recommended to have additional condition statements to further restrict authorization based on attributes that are extracted from the X.509 certificate.

When you use X.509 certificates, IAM Roles Anywhere extracts the following fields and makes them available as `PrincipalTag` elements in the session:

- `Subject`
- `Issuer`
- `Subject Alternative Name (SAN)`

These values need to match the pattern defined in STS session tags. The service ignores values that do not match this pattern. You cannot use these values in policy conditions.

> **Note**
>
> These tags are also available to be used in conditions in the identity-based policy attached to the role.

Certificate Subject fields mapping

IAM Roles Anywhere maps Relative Distinguished Names (RDNs) from an authenticating certificate's Subject into distinct `PrincipalTag` elements in the session.

The following example shows a trust policy that adds a condition based on the `Subject Common Name (CN)` of the certificate. For example, if the identity asserted in the certificate is `Alice`, a condition can be created on the `CN` of the Subject.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": "rolesanywhere.amazonaws.com"
        },
        "Action": [
          "sts:AssumeRole",
          "sts:TagSession",
          "sts:SetSourceIdentity"
        ],
        "Condition": {
          "StringEquals": {
            "aws:PrincipalTag/x509Subject/CN": "Alice"
          },
          "ArnEquals": {
            "aws:SourceArn": [
              "arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID"
            ]
          }
        }
      }
    ]
  }
```

Certificate Issuer fields mapping

IAM Roles Anywhere maps RDNs from an authenticating certificate's Issuer into distinct `PrincipalTag` elements in the session.

The following example shows a trust policy that adds a condition based on the `Issuer Common Name (CN)` of the certificate. For example, if the issuer identity asserted in the certificate is Bob, a condition can be created on the CN of the Issuer.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": "rolesanywhere.amazonaws.com"
        },
        "Action": [
          "sts:AssumeRole",
          "sts:TagSession",
          "sts:SetSourceIdentity"
        ],
        "Condition": {
          "StringEquals": {
            "aws:PrincipalTag/x509Issuer/CN": "Bob"
          },
          "ArnEquals": {
            "aws:SourceArn": [
              "arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID"
            ]
          }
        }
      }
    ]
  }
```

Certificate Subject Alternative Name (SAN) fields mapping

X.509v3 certificates may include an extension to define additional identities, call Subject Alternative Names (SANs). The SAN can have multiple values, each of which can be one of nine types – Domain Name System (DNS) names, Uniform Resource Indicators (URIs), Directory Names, IP addresses, "Other" names, EDI party names, X400 addresses, RFC 822 names, or registered OIDs. IAM Roles Anywhere will map the **first** value of the following types: `DNS Names`,

`Directory Name (DN)`, and `URI Names`. The resulting tags will be prefixed with x509SAN, with a corresponding type code. The type code will be one of DNS, URI, or Name. For values of type DN, the individual RDNs will be parsed, as with Subject and Issuer.

```
Certificate:
Data:
    Version: 3 (0x2)
    Serial Number:
        1f:71:c5:11:4a:11:9f:c0:cc:5a:5a:52:fb:37:20:ad
Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=US, O=Amazon, OU=IAM, ST=Washington, CN=RolesAnywhere, L=Seattle
    Validity
        Not Before: Apr 20 14:13:43 2022 GMT
        Not After : Jan 10 15:13:43 2023 GMT
    Subject: CN=Alice
    X509v3 extensions:
      X509v3 Subject Alternative Name:
          DNS:example.com,
          URI:spiffe://example.com/workload/alice,
          DirName:/CN=Alice
    # remainder omitted...
```

For the above certificate, the session principal tags and the SAN fields will look like the following:

```
  "aws:PrincipalTag/x509Subject/CN": "Alice",
  "aws:PrincipalTag/x509Issuer/C": "US",
  "aws:PrincipalTag/x509Issuer/O": "Amazon",
  "aws:PrincipalTag/x509Issuer/OU": "IAM",
  "aws:PrincipalTag/x509Issuer/ST": "Washington",
  "aws:PrincipalTag/x509Issuer/L": "Seattle",
  "aws:PrincipalTag/x509Issuer/CN": "RolesAnywhere",
  "aws:PrincipalTag/x509SAN/DNS": "example.com",
  "aws:PrincipalTag/x509SAN/URI": "spiffe://example.com/workload/alice",
  "aws:PrincipalTag/x509SAN/Name/CN": "Alice"
  // Additional principal tags may be available...
```

Some X.509 certificate fields can contain multiple values. For example, a Subject can have multiple Organization Unit (OU) values in the certificate. Because principal tags do not support multiple values, IAM Roles Anywhere combines multiple values into a single string, separating them with forward slashes (/) in the order they appear in the certificate.

For example, consider a certificate with these Subject values:

```
CN=alice, OU=Security, OU=Engineering, OU=Research
```

IAM Roles Anywhere creates the following principal tag:

```
"aws:PrincipalTag/x509Subject/OU": "Security/Engineering/Research"
```

This same behavior applies to any certificate field that contains multiple values. The order of values in the principal tag matches their order in the certificate.

The following example shows a trust policy that adds a condition based on the SAN fields of the certificate.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": "rolesanywhere.amazonaws.com"
        },
        "Action": [
          "sts:AssumeRole",
          "sts:TagSession",
          "sts:SetSourceIdentity"
        ],
        "Condition": {
          "StringEquals": {
            "aws:PrincipalTag/x509SAN/DNS": "example.com",
            "aws:PrincipalTag/x509SAN/URI": "spiffe://example.com/workload/
alice",
            "aws:PrincipalTag/x509SAN/Name/CN": "Alice"
          },
          "ArnEquals": {
            "aws:SourceArn": [
              "arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID"
            ]
          }
```

```
        }
      }
    ]
  }
```

> **Note**
>
> RFC 5280 allows certificates with empty subjects if and only if the SAN extension is present and marked critical. Certificates with empty subjects are NOT yet supported, since IAM Roles Anywhere uses the certificate subject as the key of the Subject resource to visualize and audit activities for certificates that are authenticated with IAM Roles Anywhere. For more information about Subject resource, see Monitoring with IAM Roles Anywhere subjects.

Using the `aws:SourceArn` or `aws:SourceAccount` or `sts:SourceIdentity` condition keys

In general, it is strongly recommended that you use the aws:SourceArn or the aws:SourceAccount global condition keys or the sts:SourceIdentity condition key in your role trust policies. This combination of conditions implements least privilege permissions and prevents IAM Roles Anywhere from acting as a potential confused deputy. When a client obtains temporary security credentials from IAM Roles Anywhere, the `aws:SourceArn` and `aws:SourceAccount` will be set based on the ARN of the trust anchor specified in the call to `CreateSession`.

To use the `aws:SourceArn` and `aws:SourceAccount` global condition keys, set the value to the Amazon Resource Name (ARN) or account of the trust anchor that you expect to use for authentication. As an example, consider the following role trust policy.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": "rolesanywhere.amazonaws.com"
        },
        "Action": [
```

```
        "sts:AssumeRole",
        "sts:TagSession",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID"
          ]
        }
      }
    }
  ]
}
```

To use the `sts:SourceIdentity` condition key, set the source identity prefix and source identity value according to the following rules. As an example, consider the following role trust policy.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "rolesanywhere.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": {
          "sts:SourceIdentity": [
            "${sourceIdentityPrefix}${sourceIdentityValue}"
          ]
        },
        "ArnEquals": {
          "aws:SourceArn": [
```

```
            "arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "rolesanywhere.amazonaws.com"
      },
      "Action": [
        "sts:TagSession"
      ]
    }
  ]
}
```

Certificate attribute mapping

IAM Roles Anywhere provides you with the capability to define a custom set of mapping rules, enabling you to specify which data are extracted from authenticating certificates as session tags for authorization policies. These customized attribute mappings are associated with a profile.

Attributes are data elements that come from specific fields in the certificate. You can use specifiers to represent one or more attributes.

> **Note**
>
> For information about session tag quotas, see Session tagging operations.

Topics

- Default mapping behavior
- Put attribute mappings
- Delete attribute mappings
- Attribute mapping and trust policy

Default mapping behavior

The following attributes are mapped by default when you create a profile. The default mapping rules are as follows:

- `x509Subject`: maps all supported Relative Distinguished Names (RDNs) from an authenticating certificate's Subject into distinct `PrincipalTag` elements in the session.
- `x509Issuer`: maps all supported Relative Distinguished Names (RDNs) from an authenticating certificate's Issuer into distinct `PrincipalTag` elements in the session.
- `x509SAN (Subject Alternative Name)`: maps the **first** value of the following types: `DNS Names`, `Directory Name (DN)`, and `URI Names`

To view your current mappings associated with a profile, using the following command:

```
$aws rolesanywhere get-profile --profile-id PROFILE_ID
```

Default mapping rules in a JSON format:

```
"attributeMappings": [
  {
    "mappingRules": [
        {
            "specifier": "*"
        }
    ],
    "certificateField": "x509Issuer"
  },
  {
    "mappingRules": [
        {
            "specifier": "DNS"
        },
        {
            "specifier": "URI"
        },
        {
            "specifier": "Name/*"
        }
    ],
    "certificateField": "x509SAN"
```

```
  },
  {
    "mappingRules": [
        {
            "specifier": "*"
        }
    ],
    "certificateField": "x509Subject"
  }
]
```

Note

If you see * as a specifier, it signifies the default behavior, which maps all recognizable RDNs for `x509Subject`, `x509Issuer` and `x509SAN/Name`. However, * does not have a defined behavior in the context of `x509SAN/URI`, `x509SAN/DNS`, or `x509SAN/`. The specifier `Name/` represents the first recognizable attribute of the `Directory Name`. Both `Name` and `Name/` are equivalent to `Name/*` and will be displayed as `Name/*`in the mapping rule.

Put attribute mappings

Put attribute mappings (command line interface)

`put-attribute-mapping` enables you to attach new mapping rules to your profile. When using that profile, the certificate mapping behavior changes according to your customized rules.

To put a mapping rule, using the following command:

```
$aws rolesanywhere put-attribute-mapping \
        --certificate-field CERTIFICATE_FIELD \
        --mapping-rules specifier=SPECIFIER \
        --profile-id PROFILE_ID
```

The `CERTIFICATE_FIELD` can be in one of `x509Subject`, `x509Issuer` and `x509SAN`. The `SPECIFIER` is a string enforced by a standard (for example, OID) that can map to a piece of information encoded in the certificate.

For example, to add mapping rules for `x509Subject/CN` and `x509Subject/OU`, use the following command:

```
$aws rolesanywhere put-attribute-mapping \
        --certificate-field x509Subject \
        --mapping-rules specifier=CN specifier=OU \
        --profile-id PROFILE_ID
```

Put attribute mappings (console)

1. Sign in to IAM Roles Anywhere console.
2. Scroll to find profile table and **choose the profile** to add certificate attribute mappings.
3. Within profile detail page scroll towards **Certificate attribute mappings** section and choose **Manage mappings**.
4. Scroll to find the **Add mappings** button and click on it.
5. Choose a certificate field from either `Subject`, `Issuer`, or `Subject Alternative Name` in the dropdown list, and enter the specifier
6. Select **Save changes** to add attribute mappings.

Delete attribute mappings

Delete attribute mappings (command line interface)

`delete-attribute-mapping` enables you to delete mapping rules from your profile. When using that profile, the attribute specified by the deleted mapping rule will not be mapped from a certificate.

To delete a mapping rule, using the following command:

```
$aws rolesanywhere delete-attribute-mapping \
        --certificate-field CERTIFICATE_FIELD \
        --specifiers SPECIFIERS \
        --profile-id PROFILE_ID
```

The `CERTIFICATE_FIELD` can be in one of `x509Subject`, `x509Issuer` and `x509SAN`. The `SPECIFIER` is a string enforced by a standard (for example, OID) that exists in your current mapping rules.

For example, to delete mapping rules for `x509Subject/CN` and `x509Subject/OU`, use the following command:

```
$aws rolesanywhere delete-attribute-mapping \
        --certificate-field x509Subject \
        --specifiers CN OU \
        --profile-id PROFILE_ID
```

Delete attribute mappings (console)

1. Sign in to IAM Roles Anywhere console.
2. Scroll to find profile table and **choose the profile** to remove certificate attribute mappings.
3. Within profile detail page scroll towards **Certificate attribute mappings** section and choose **Manage mappings**.
4. Scroll to find the corresponding attribute mapping row and click on **Remove mapping** button associated with it.
5. Select **Save changes** to remove attribute mappings.

Attribute mapping and trust policy

It is recommended to have condition statements in the Assume Role Policy Document to restrict authorization based on attributes that are extracted from an end-entity X.509 certificate. For more information about the role trust policy, see Trust policy.

The attribute mapping field of a profile controls which attributes from an authenticating X.509 certificate will be mapped for principal tags. Therefore, while adding condition statements to an Assume Role Policy Document, be cautious that the specifiers used in mapping rules for authorization need to be mapped accordingly.

The following example shows trust policies that add a condition based on the `Issuer Common Name (CN)` of the certificate.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "rolesanywhere.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/x509Issuer/CN": "Bob"
        },
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID"
          ]
        }
      }
    }
  ]
}
```

If a profile is used with an Attribute Mapping field that lacks `specifier: CN` or `specifier: *` in the mappingRules for `x509Issuer`, the first condition in the Assume Role Policy Document will evaluate as `false` because there will be no value mapped `aws:PrincipalTag/x509Issuer/CN`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "rolesanywhere.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession",
```

```
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalTag/x509Issuer/CN": "Bob"
        },
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID"
          ]
        }
      }
    }
  ]
}
```

Likewise, if the condition is `StringNotEquals`, the condition will evaluate to `true` using the same profile. This happens because the condition is disregarded when the principal tag is dropped due to attribute mapping APIs.

Having the Attribute Mapping field provided below in a profile, the `StringEquals` condition for `x509Issuer/CN` will assess to `false`, or the `StringNotEquals` condition will assess to `true`.

```
"attributeMappings": [
  {
    "mappingRules": [
        {
            "specifier": "O"
        }
    ],
    "certificateField": "x509Issuer"
  },
  {
    "mappingRules": [
        {
            "specifier": "DNS"
        },
        {
            "specifier": "URI"
        },
        {
            "specifier": "Name/*"
```

```
        }
    ],
    "certificateField": "x509SAN"
  },
  {
    "mappingRules": [
        {
            "specifier": "*"
        }
    ],
    "certificateField": "x509Subject"
  }
]
```

Source identity rules

We define a source identity prefix as follows:

- `"CN="`: the common name of the subject in the certificate is set and less than or equal to 61 characters.
- `"ID="`: the common name of the subject in the certificate is not set. This value is left-padded with zero to be even in length.
- `""`: (empty string) the common name of the subject in the certificate is set and has a length from 62 to 64 characters.

Hex encoding example:

- Decimal serial number 291 converts to hex 123, which becomes `ID=0123`
- Decimal serial number 17767 converts to hex 4567, which becomes `ID=4567`

Revocation

Certificate revocation is supported through the use of imported certificate revocation lists (CRLs). Currently, certification revocation is only supported by the API and CLI. You can import a CRL that is generated from your CA using `ImportCrl` API or `import-crl` CLI command. Certificates used for authentication will be checked for their revocation status.

Callbacks to CRL Distribution Points (CDPs) or Online Certificate Status Protocol (OCSP) endpoints are not supported.

The IAM Roles Anywhere authentication process

To provide credentials, AWS Identity and Access Management Roles Anywhere uses the IAM Roles Anywhere CreateSession API. The API authenticates requests with a signature using keys associated with the X.509 certificate, which was used for authentication. It acts like `AssumeRole` – exchanging the signature for a standard SigV4-compatible session credential.

To successfully authenticate, the following constraints must be satisfied:

- The signature attached to the request **MUST** be validated against the signing certificate (also attached to the request).
- The signing certificate **MUST** have a valid trust chain to a Certificate Authority (CA) certificate configured in the customer account.
- The target role for which credentials are issued **MUST** have an `AssumeRolePolicyDocument` that allows IAM Roles Anywhere service principal, `rolesanywhere.amazonaws.com`, to call `sts:AssumeRole`, `sts:TagSession`, and `sts:SetSourceIdentity`. For more information, see Granting permissions to pass a role to a service in the *IAM User Guide*.
- The target role for which credentials are issued **MAY** have additional `Condition` predicates in the `AssumeRolePolicyDocument` that restrict authorization based on attributes extracted from the X.509 Certificate (for example, Subject or Issuer).

The signature uses the same canonicalization mechanism as AWS Signature V4 for API requests (SigV4), with the following changes and additions:

- The private key used to sign the request **MUST** be bound to an X.509 Certificate.
- The signing certificate **MUST** be a v3 certificate.
- The signing certificate **MUST** be attached to the request via the header `X-Amz-X509`, as Base64-encoded Distinguished Encoding Rules (DER) data.
- The relevant headers – `X-Amz-X509` and `X-Amz-X509-Chain` (if applicable) **MUST** be included in the signed headers field of the `Authorization` header.
- The `X-Amz-X509-Chain` header **MUST** be encoded as comma-delimited, base64-encoded DER.
- The `X-Amx-X509-Chain` header **MUST NOT** exceed the maximum depth of 5 certificates.
- The signing certificate's serial number **MUST** be included in the Credential portion of the Scope field of the `Authorization` header.

RSA and EC keys are supported; RSA keys are used with the RSA PKCS#1 v1.5 signing algorithm. EC keys are used with the ECDSA.

Topics

- IAM Roles Anywhere CreateSession API
- The IAM Roles Anywhere authentication signing process

IAM Roles Anywhere CreateSession API

The CreateSession API returns temporary security credentials for workloads that have been authenticated with IAM Roles Anywhere to access AWS resources. For endpoints, see: Roles Anywhere Endpoints and Quotas

Request Syntax

```
POST /sessions HTTP/1.1
Content-type: application/json
{
  "durationSeconds": number,
  "profileArn": string,
  "roleArn": string,
  "trustAnchorArn": string,
  "roleSessionName": string,
}
```

URI Request Parameters

The request accepts the following as URI parameters or as JSON in the request body. Our examples put these in the request body.

`profileArn`

The Amazon Resource Name (ARN) of the profile.

Type: String

Required: Yes

`roleArn`

The Amazon Resource Name (ARN) of the role to assume.

Type: String

Required: Yes

`trustAnchorArn`

The Amazon Resource Name (ARN) of the trust anchor.

Type: String

Required: Yes

Request Body

The request accepts the following data in JSON format.

`durationSeconds`

The duration, in seconds, of the role session. The value is optional and can range from 900 seconds (15 minutes) up to 43200 seconds (12 hours). Please see the `Expiration` subsection of the Credentials Object section for more details on how this value is used in determining the expiration of the vended session.

Type: Number

Required: No

`profileArn`

The Amazon Resource Name (ARN) of the profile.

Type: String

Required: Yes

`roleArn`

The Amazon Resource Name (ARN) of the role to assume.

Type: String

Required: Yes

`sessionName`

This parameter has been deprecated.

This parameter is no longer used. Instead, use the `roleSessionName` parameter.

Type: String

Required: No

`trustAnchorArn`

The Amazon Resource Name (ARN) of the trust anchor.

Type: String

Required: Yes

`roleSessionName`

An identifier for the role session.

Type: String

Required: No

Response Syntax

```
HTTP/1.1 201
Content-type: application/json
{
  "credentialSet":[
    {
      "assumedRoleUser": {
      "arn": ARN,
      "assumedRoleId": String
      },
      "credentials":{
        "accessKeyId": String,
        "expiration": Timestamp,
        "secretAccessKey": String,
        "sessionToken": String
      },
      "packedPolicySize": Number,
```

```
        "roleArn": ARN,
        "sourceIdentity": String
      }
    ],
    "subjectArn": ARN
  }
```

Response Elements

If the action is successful, the service sends back an HTTP 201 response.

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

`assumedRoleUser`

The Amazon Resource Name (ARN) and the assumed role ID, which are identifiers that you can use to refer to the resulting temporary security credentials.

Type: AssumedRoleUser object

`credentials`

The temporary security credentials, which include an access key ID, a secret access key, and a security (or session) token.

Type: Credentials Object

`packedPolicySize`

A percentage value that indicates the packed size of the session policies and session tags combined passed in the request. The request fails if the packed size is greater than 100 percent, which means the policies and tags exceeded the allowed space.

Type: Integer

Valid Range: Minimum value of 0

`sourceIdentity`

The source identity is specified by the principal that is calling the CreateSession operation. For more information on how the source identity is derived please see the Trust Model section.

Type: String

Pattern: `[\p{L}\p{Z}\p{N}_.:/=+\-@]+`

`roleArn`

The Amazon Resource Name (ARN) of the assumed role.

Type: String

`subjectArn`

The Amazon Resource Name (ARN) of the Subject resource.

The Subject resource records the history of the principal that is calling the CreateSession operation, including its first recorded authentication time, and last recorded authentication time.

Type: String

Credentials Object

AWS credentials for API authentication.

`AccessKeyId`

The access key ID that identifies the temporary security credentials.

Type: String

Length Constraints: Minimum length of 16. Maximum length of 128.

Required: Yes

`Expiration`

The time at which the vended credentials expire. This value is determined based on the values set for `profileDurationSeconds` and `createSessionDurationSeconds`, where `profileDurationSeconds` is the value of the `durationSeconds` field that's set on the profile referenced in the call to `CreateSession`, and `createSessionDurationSeconds` is the value of the `durationSeconds` parameter in the request to `CreateSession`. From this, `finalDurationSeconds` is determined by the following: `finalDurationSeconds = min(profileDurationSeconds, createSessionDurationSeconds)`, where `finalDurationSeconds` is the value for `durationSeconds` that will be sent in the `AssumeRole` request to assume the target role. If `createSessionDurationSeconds` isn't provided, then `finalDurationSeconds = profileDurationSeconds`. From this, `Expiration` is determined by the following: `Expiration = CurrentTime +`

`finalDurationSeconds`. Note that if `finalDurationSeconds` is greater than the maximum session duration (`MaxSessionDuration`) set on the role, you will receive an error response from the API.

Type: Timestamp

Required: Yes

`SecretAccessKey`

The secret access key that can be used to sign requests.

Type: String

Required: Yes

`SessionToken`

The token that users must pass to the service API to use the temporary credentials.

Type: String

Required: Yes

The relationship between CreateSession and AssumeRole

CreateSession is an X.509 wrapper around AssumeRole. The temporary session credentials are delivered to RolesAnywhere by AssumeRole, and then passed on without modification in the result of CreateSession. CreateSession is not included in any SDK or client as there is not yet native SDK or client support for CreateSession's signing process.

The `roleSessionName` option in the CreateSession request allows you to customize the identifier of a role session. The provided value will be passed into the AssumeRole request during the CreateSession process and will be incorporated into the ARN and ID of the assumed role user. This means that subsequent API requests using the temporary session credentials vended by CreateSession will expose the role session name to the corresponding account in their CloudTrail logs. Also, you can reference these credentials in a resource-based policy by the ARN or the ID. The default role session name, when not provided, is the serial number of the X.509 certificate provided in a session request.

The `acceptRoleSessionName` option on a profile controls whether or not a role session name will be accepted in a session request with this profile. By setting the `acceptRoleSessionName`

option to `true`, you acknowledge that a role session name will be honored in a session request if the CreateSession call is made with this specific profile. This means that the default role session name, which is the serial number of the X.509 certificate, will no longer be used and included in the ARN or ID of the assumed role user.

By default, the `acceptRoleSessionName` attached to the profile is `false`. If you provide a custom role session name in the CreateSession request but custom role session names are not accepted, you will receive an `Access Denied` error. However, when a role session name is not presented in the CreateSession request, the role session name will default to the serial number of the end-entity X.509 certificate that was used to authenticate the request, even when the `acceptRoleSessionName` on the profile is `true`.

The IAM Roles Anywhere authentication signing process

The signing process is identical to SigV4, with the exception of the keys used, the signature algorithm, and the addition of headers related to the X.509 certificate and trust chain. For more information, see AWS Signature Version 4 for API requests, which should be treated as authoritative unless specifically addressed in this user guide.

Topics

- Task 1: Create a canonical request
- Task 2: Create a string to sign
- Task 3: Calculate the signature
- Task 4: Add the signature to the HTTP request

Task 1: Create a canonical request

To begin the signing process, create a string that includes information from your request in a standardized (canonical) format. This ensures that when the request is received, the string to sign can be calculated in the same way you did. This string will then be used for signature verification by IAM Roles Anywhere.

Follow the steps below to create a canonical version of the request. Otherwise, your version and the version calculated by AWS won't match, and the request will be denied.

The following example shows the pseudocode to create a canonical request.

Example Canonical request pseudocode

```
CanonicalRequest =
  HttpRequestMethod + '\n' +
  CanonicalUri + '\n' +
  CanonicalQueryString + '\n' +
  CanonicalHeaders + '\n' +
  SignedHeaders + '\n' +
  Lowercase(Hex(SHA256(RequestPayload)))
```

A canonical request has the above structure, in which specific elements of the request are transformed into a canonical value and concatenated together with other elements, joined with a newline character.

The following examples shows how to construct the canonical form of a request to IAM Roles Anywhere. The original request might look like this as it is sent from the client to AWS.

Example Request

```
POST /sessions HTTP/1.1
Host: rolesanywhere.us-east-1.amazonaws.com
Content-Type: application/json
X-Amz-Date: 20211103T120000Z
X-Amz-X509: {base64-encoded DER data}
{
  "durationSeconds": number,
  "profileArn": string,
  "roleArn": string,
  "trustAnchorArn": string
}
```

To create a canonical request, concatenate the following components from each step into a single string:

1. The `HttpRequestMethod` is the verb of the HTTP request, in upper case. From the example, POST.

2. The `CanonicalUri` is the path of the request up until the query string delimiter ?. From the example, `/sessions`. The path MUST be normalized according to RFC 3986, with redundant and relative path components removed. Path segments MUST be URI-encoded twice.

3. The `CanonicalQueryString` is the string following ? in the request URI. If the request does not include a query string, use an empty string (essentially, a blank line). The example request does not include a query string, therefore we will use an empty string. To construct the canonical query string, complete the following steps:

   a. Sort the parameter names by character code point in ascending order. Parameters with duplicate names should be sorted by value. For example, a parameter name that begins with the uppercase letter F precedes a parameter name that begins with a lowercase letter b.

   b. URI-encode each parameter name and value according to the following rules:

      i. Do not URI-encode any of the unreserved characters that RFC 3986 defines: A-Z, a-z, 0-9, hyphen (-), underscore (_), period (.), and tilde (~).

      ii. Percent-encode all other characters with %XY, where X and Y are hexadecimal characters (0-9 and uppercase A-F). For example, the space character must be encoded as %20 (not using '+', as some encoding schemes do) and extended UTF-8 characters must be in the form %XY%ZA%BC.

      iii. Double-encode any equals (=) characters in parameter values.

   c. Build the canonical query string by starting with the first parameter name in the sorted list.

   d. For each parameter, append the URI-encoded parameter name, followed by the equals sign character (=), followed by the URI-encoded parameter value. Use an empty string for parameters that have no value.

   e. Append the ampersand character (&) after each parameter value, except for the last value in the list.

4. The `CanonicalHeaders` is a string capturing the header key and value that are included in the signature. The field is structured as follows:

```
   CanonicalHeaders =
     CanonicalHeadersEntry0 + CanonicalHeadersEntry1 + ... +
CanonicalHeadersEntryN
   CanonicalHeadersEntry =
```

```
    Lowercase(HeaderName) + ':' + Trimall(HeaderValue) + '\n'
```

Lowercase represents a function that converts all characters to lowercase. The `TrimAll` function removes excess white space before and after values, and converts sequential spaces to a single space.

> **Important**
>
> The signing certificate MUST be presented in the header `X-Amz-X509`, as base64-encoded Distinguished Encoding Rules (DER), and the `X-Amz-X509` header MUST be included in `CanonicalHeaders` and `SignedHeaders`
> If the client is providing the chain of intermediate certificates, the `X-Amz-X509-Chain` MUST be added to the request as well.

Build the canonical headers list by sorting the (lowercase) headers by character code and then iterating through the header names. Construct each header according to the following rules:

- Append the lowercase header name followed by a colon.
- Append a comma-separated list of values for that header. Do not sort the values in headers that have multiple values.
- Append a new line ('\n').

5. The `SignedHeaders` is a list of the header names, sorted by lowercase character code, delimited by semi-colon. For example – `content-type;host;x-amz-date;x-amz-x509`
6. A hash of the request payload is appended to the canonical request. The bytes of the request are encoded as UTF-8, hashed with SHA-256, the resulting bytes hex encoded, and finally lowercased.
7. To construct the finished canonical request, combine all the components from each step as a single string. As noted, each component ends with a newline character. If you follow the canonical request pseudocode explained earlier, the resulting canonical request is shown in the following example.

Example Canonical request

```
POST
/sessions
```

```
content-type:application/json
host:rolesanywhere.us-east-1.amazonaws.com
x-amz-date:20211103T120000Z
x-amz-x509:{base64-encoded DER data}

content-type;host;x-amz-date;x-amz-x509
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

8. Create a digest (hash) of the canonical request with the same algorithm that you used to hash the payload. The bytes of the request are encoded as UTF-8, hashed with SHA-256, the resulting bytes hex encoded, and finally lowercased.

Task 2: Create a string to sign

The “string to sign” is the actual input to the signing algorithm, and includes meta information about the request, along with the canonical request created in the previous step.

Example StringToSign pseudocode

```
StringToSign =
  Algorithm + '\n' +
  RequestDateTime + '\n' +
  CredentialScope + '\n' +
  HashedCanonicalRequest
```

The structure is as follows:

1. The `Algorithm` is a string that indicates how the signature is calculated. It is of the form `AWS4-X509-[AlgorithmId]-[HASH]`, where `AlgorithmId` is either `RSA` or `ECDSA`, and MUST be supported by the key type of the signing certificate's private key. For example, if the signing certificate has an RSA private key, the full algorithm string will be `AWS4-X509-RSA-SHA256`.
2. The `RequestDateTime` is a string derived at time of the signing operation, at second granularity, in UTC, formatted as ISO8601 basic, YYYYMMDD’T’HHMMSS’Z’. For example, `20211101T121030Z`.

3. The CredentialScope is structured field of the form `Date + '/' + Region + '/' + Service + '/aws4_request'`. The Region and service name strings must be UTF-8 encoded. For example, `20211101/us-east-1/rolesanywhere/aws4_request`.
4. Finally, append a newline followed by the HashedCanonicalRequest computed in the previous step.

Task 3: Calculate the signature

```
Signature = HexEncode(SigningAlgorithm(CertPrivateKey,StringToSign))
```

Here, the SigningAlgorithm refers to a supported signing operation, either SHA256WithRSA or SHA256WithECDSA.

Task 4: Add the signature to the HTTP request

The signature derived from the previous step is added to the HTTP request in the `Authorization` header field. The `Authorization` header is attached to the request and validated for authentication. It is partitioned into multiple fields – signing algorithm, credentials, signed headers, and the actual signature. The header of the authentication mechanism based on X.509 differs from a SigV4 header in two ways:

- Algorithm. As described above, instead of `AWS4-HMAC-SHA256`, the algorithm field will have the values of the form `AWS4-X509-RSA-SHA256` or `AWS4-X509-ECDSA-SHA256`, depending on whether an RSA or Elliptic Curve algorithm is used. This, in turn, is determined by the key bound to the signing certificate.
- Scope field/credentials. As specified above, the serial number of the certificate used to sign the request will be in place of the Access Key ID (credential) in the Scope field.

The structure of the field is as follows:

```
    Authorization: {Algorithm} Credential={CredentialString},
SignedHeaders={SignedHeaders}, Signature={Signature}
```

1. The `Algorithm` is of the form `AWS4-X509-{RSA|ECDSA}-SHA256`. Examples – `AWS4-X509-RSA-SHA256`.
2. The `Credential` is constructed via `{SerialNumber}/{Scope}` where serial number is the decimal representation of the serial number of the signing certificate, and `Scope` is the value constructed as input to the `StringToSign`. For example:

   ```
   Credential=111112222233333344444/20201105/us-east-1/rolesanywhere/aws4_request
   ```

3. The `SignedHeaders` is a comma-delimited list of the headers signed as part of the request.
4. The `Signature` is the hex encoded output of the previous step.

IAM Roles Anywhere cloud security and shared responsibility

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 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 AWS Identity and Access Management Roles Anywhere, 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 Identity and Access Management Roles Anywhere. The following topics show you how to configure IAM Roles Anywhere to meet your security and compliance objectives. You can also learn how to use other AWS services that help you to monitor and secure your IAM Roles Anywhere resources.

Topics

- Mapping identities to your workloads with IAM Roles Anywhere
- Data protection and privacy in IAM Roles Anywhere
- Identity and access management for IAM Roles Anywhere
- Disaster recovery and resilience in IAM Roles Anywhere
- IAM Roles Anywhere and interface VPC endpoints (AWS PrivateLink)
- Control API access with IAM policies

Mapping identities to your workloads with IAM Roles Anywhere

A key element to using IAM Roles Anywhere is managing how identities are assigned to workloads. Certificates are issued to compute instances (servers, containers), in which the identity is encoded as the `Subject` of the certificate. The subject may be a simple Common Name (CN), a Fully Qualified Distinguished Name (FQDN), that contains information about organizational structure, or a simple hostname. Alternatively, a standard such as SPIFFE could be used to create a hierarchical namespace for the workload identities.

IAM Roles Anywhere lets the workload use the certificate to obtain temporary credentials instead of issuing Access Key IDs and Secret Access Keys, and the identity in the subject is encoded in the session credentials in a way that can be used in resource policies. For example, if a certificate has a `Subject` with `CN=Alice`, the value is added to the session as a `PrincipalTag`: `aws:PrincipalTag/x509Subject/CN`.

The fields Subject, Issuer and Subject Alternative Name (SAN) are extracted from x509 tickets and used as elements of the PrincipalTags. Tags that start with the prefix x509Subject are usually followed by the suffix /CN used to identify the subject's common name. Tags starting with the prefix x509Issuer are usually followed by /C, /O, /OU, /ST, /L, and /CN in order to identify the issuer's country, organization, organization unit, state, locality and common name respectively. Tags starting with x509SAN prefix are followed by /DNS, /URI or /CN to identify the subject alternative name's DNS, URI or common name of the subject alternative name respectively. These are some of the different ways x509 fields are implemented as PrincipalTags for use in identity mapping.

Data protection and privacy in IAM Roles Anywhere

The AWS shared responsibility model applies to data protection in AWS Identity and Access Management Roles Anywhere. 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 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 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 IAM Roles Anywhere 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.

Encryption in transit

IAM Roles Anywhere provides secure and private endpoints for encrypting data in transit. The secure and private endpoints allow AWS to protect the integrity of API requests to IAM Roles Anywhere. AWS requires API calls to be signed by the caller using a secret access key. This requirement is stated in the Signature Version 4 Signing Process (Sigv4).

Key management

Authenticating to IAM Roles Anywhere requires the use of asymmetric (public/private) key pairs. IAM Roles Anywhere does not hold customer private keys. Those remain on customer instances in data centers. Care should be taken to minimize the risk of accidental disclosure.

- Use OS file system permissions to restrict read access to the owning user.

- Never check keys into source control. Store them separately from source code to reduce the risk of accidentally including them in a change set. If possible, consider the use of a secure storage mechanism.

Inter-network traffic privacy

When accessing AWS APIs and resources from your data center, be aware of how the traffic is routed. Connectivity may occur via Network Address Translation (NAT) at the data center outbound firewall, or it may occur through Virtual Private Network (VPN).

Identity and access management for IAM Roles Anywhere

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 IAM Roles Anywhere resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- How IAM Roles Anywhere works with IAM
- Identity-based policy examples for IAM Roles Anywhere
- Troubleshooting IAM Roles Anywhere identity and access
- Using service-linked roles for IAM Roles Anywhere
- AWS managed policies for IAM Roles Anywhere

Audience

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

Service user – If you use the IAM Roles Anywhere service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more IAM Roles Anywhere 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 IAM Roles Anywhere, see Troubleshooting IAM Roles Anywhere identity and access.

Service administrator – If you're in charge of IAM Roles Anywhere resources at your company, you probably have full access to IAM Roles Anywhere. It's your job to determine which IAM Roles Anywhere 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 IAM Roles Anywhere, see How IAM Roles Anywhere works with IAM.

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

Authenticating with identities

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

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might 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 Multi-factor authentication in the *AWS IAM Identity Center User Guide* and AWS Multi-factor authentication in IAM 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 don't 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 *IAM User Guide*.

Federated identity

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

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

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 Use cases for IAM users 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. To temporarily assume an IAM role in the AWS Management Console, you can switch from a user to an IAM role (console). 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 Methods to assume a role in the *IAM User Guide*.

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

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Create a role for a third-party identity provider (federation) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **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 Cross account resource access in IAM 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.

- **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
- **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 Create 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 AWS 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 Use an IAM role to grant permissions to applications running on Amazon EC2 instances 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**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

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 Define custom IAM permissions with customer managed 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 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 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 an 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 Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see Resource control policies (RCPs) 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 IAM Roles Anywhere works with IAM

Before you use IAM to manage access to IAM Roles Anywhere, learn what IAM features are available to use with IAM Roles Anywhere.

IAM features you can use with AWS Identity and Access Management Roles Anywhere

IAM feature	IAM Roles Anywhere 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)	Partial
Temporary credentials	Yes
Principal permissions	Yes
Service roles	Yes
Service-linked roles	Yes

To get a high-level view of how IAM Roles Anywhere 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 IAM Roles Anywhere

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. 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 IAM Roles Anywhere

To view examples of IAM Roles Anywhere identity-based policies, see Identity-based policy examples for IAM Roles Anywhere.

Resource-based policies within IAM Roles Anywhere

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 Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for IAM Roles Anywhere

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 IAM Roles Anywhere actions, see Actions defined by AWS Identity and Access Management Roles Anywhere in the *Service Authorization Reference*.

Policy actions in IAM Roles Anywhere use the following prefix before the action:

```
rolesanywhere
```

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

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

To view examples of IAM Roles Anywhere identity-based policies, see Identity-based policy examples for IAM Roles Anywhere.

Policy resources for IAM Roles Anywhere

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": "*"
```

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

To view examples of IAM Roles Anywhere identity-based policies, see Identity-based policy examples for IAM Roles Anywhere.

Policy condition keys for IAM Roles Anywhere

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*.

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

To view examples of IAM Roles Anywhere identity-based policies, see Identity-based policy examples for IAM Roles Anywhere.

Access control lists (ACLs) in IAM Roles Anywhere

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 (ABAC) with IAM Roles Anywhere

Supports ABAC (tags in policies): Partial

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.

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 IAM Roles Anywhere

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 Switch from a user to an IAM 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 IAM Roles Anywhere

Supports forward access sessions (FAS): Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

Service roles for IAM Roles Anywhere

Supports service roles: Yes

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 IAM Roles Anywhere functionality. Edit service roles only when IAM Roles Anywhere provides guidance to do so.

Service-linked roles for IAM Roles Anywhere

Supports service-linked roles: Yes

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 IAM Roles Anywhere

By default, users and roles don't have permission to create or modify IAM Roles Anywhere resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

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 IAM Roles Anywhere, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for AWS Identity and Access Management Roles Anywhere in the *Service Authorization Reference*.

Topics

- Policy best practices
- Using the IAM Roles Anywhere console
- Allow users to view their own permissions

Policy best practices

Identity-based policies determine whether someone can create, access, or delete IAM Roles Anywhere 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 AWS 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*.

Using the IAM Roles Anywhere console

To access the AWS Identity and Access Management Roles Anywhere console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the IAM Roles Anywhere resources in 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 (users or roles) with that policy.

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, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that users and roles can still use the IAM Roles Anywhere console, also attach the IAM Roles Anywhere `ConsoleAccess` or `ReadOnly` AWS managed policy to the entities. For more information, see Adding permissions to a user 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": "*"
        }
    ]
}
```

Troubleshooting IAM Roles Anywhere identity and access

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

Topics

- I am not authorized to perform an action in IAM Roles Anywhere
- I am not authorized to perform iam:PassRole
- I want to view my access keys
- I'm an administrator and want to allow others to access IAM Roles Anywhere
- I want to allow people outside of my AWS account to access my IAM Roles Anywhere resources
- Invalidating a IAM Roles Anywhere session

I am not authorized to perform an action in IAM Roles Anywhere

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 a fictional *`my-example-widget`* resource but does not have the fictional `rolesanywhere:`*`GetWidget`* permissions.

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

In this case, Mateo asks his administrator to update his policies to allow him to access the *`my-example-widget`* resource using the `rolesanywhere:`*`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 IAM Roles Anywhere.

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 IAM Roles Anywhere. 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 AWS 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 IAM Roles Anywhere

To allow others to access IAM Roles Anywhere, you must grant permission to the people or applications that need access. If you are using AWS IAM Identity Center to manage people and applications, you assign permission sets to users or groups to define their level of access. Permission sets automatically create and assign IAM policies to IAM roles that are associated with the person or application. For more information, see Permission sets in the *AWS IAM Identity Center User Guide*.

If you are not using IAM Identity Center, you must create IAM entities (users or roles) for the people or applications that need access. You must then attach a policy to the entity that grants them the correct permissions in IAM Roles Anywhere. After the permissions are granted, provide the credentials to the user or application developer. They will use those credentials to access AWS. To learn more about creating IAM users, groups, policies, and permissions, see IAM Identities and Policies and permissions in IAM in the *IAM User Guide*.

I want to allow people outside of my AWS account to access my IAM Roles Anywhere 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 IAM Roles Anywhere supports these features, see How IAM Roles Anywhere 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*.

Invalidating a IAM Roles Anywhere session

You can invalidate a IAM Roles Anywhere session if you need to revoke access for certificates issued by AWS Private Certificate Authority (AWS Private CA) or another certificate authority.

To invalidate a IAM Roles Anywhere session

1. To get an updated certificate revocation list (CRL), do one of the following:
   - If you use AWS Private CA, see Revoking IAM role session permissions.
   - If you use a different certificate authority:
     a. Follow their documentation for revoking certificate access and invalidating sessions.
     b. Request a new CRL from your certificate authority.
2. After you get your updated CRL, import it using one of the following:
   - The ImportCrl API operation

- The import-crl AWS CLI command

Using service-linked roles for IAM Roles Anywhere

AWS Identity and Access Management Roles Anywhere uses IAM service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to IAM Roles Anywhere. Service-linked roles are predefined by IAM Roles Anywhere and include all the permissions that the service requires to publish CloudWatch metrics and ensure the private certificate authorities you use as trust anchors can be accessed as part of authenticating with IAM Roles Anywhere. They also ensure that you receive auditing information regarding IAM Roles Anywhere. Service-linked roles are different from the roles that you configure for the service and obtain temporary credentials for.

A service-linked role makes setting up IAM Roles Anywhere easier because you don't have to manually add the necessary permissions. IAM Roles Anywhere defines the permissions of its service-linked roles. Unless defined otherwise, only IAM Roles Anywhere can assume its service-linked roles. The defined permissions include the trust policy and the permissions policy. The permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting its related resources. This protects your IAM Roles Anywhere resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS services that work with IAM and look for the services that have **Yes** in the **Service-linked roles** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for IAM Roles Anywhere

IAM Roles Anywhere uses the service-linked role named **AWSServiceRoleForRolesAnywhere** which allows IAM Roles Anywhere to publish CloudWatch metrics and check the configuration of AWS Private CA in your account. This service-linked role has an IAM policy attached to it named AWSRolesAnywhereServicePolicy.

The AWSServiceRoleForRolesAnywhere service-linked role trusts the following services to assume the role:

- `rolesanywhere.amazonaws.com`

The role permissions policy named AWSRolesAnywhereServicePolicy allows IAM Roles Anywhere to complete the following actions on the specified resources:

- Actions on CloudWatch:
  - `cloudwatch:PutMetricData` – Allows IAM Roles Anywhere to publish metric data points to the `AWS/RolesAnywhere` and `AWS/Usage` namespaces.
- Actions on AWS Private CA:
  - `acm-pca:GetCertificateAuthorityCertificate` – Allows IAM Roles Anywhere to retrieve the certificate and certificate chain for your private certificate authority.
  - `acm-pca:DescribeCertificateAuthority` – Allows IAM Roles Anywhere to list information about your private certificate authority.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "cloudwatch:namespace": [
            "AWS/RolesAnywhere",
            "AWS/Usage"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "acm-pca:GetCertificateAuthorityCertificate",
        "acm-pca:DescribeCertificateAuthority"
      ],
```

```
      "Resource": "arn:aws:acm-pca:*:*:*"
    }
  ]
}
```

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "cloudwatch:namespace": [
            "AWS/RolesAnywhere",
            "AWS/Usage"
          ]
        }
      }
    }
  ]
}
```

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-linked role permissions in the *IAM User Guide*.

Creating a service-linked role for IAM Roles Anywhere

You don't need to manually create a service-linked role. When you create your first trust anchor in the AWS Management Console, the AWS CLI, or the AWS API, IAM Roles Anywhere creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. Note that credentials will still be issued, but metrics will not be reported.

You can also use the IAM console to create a service-linked role when you have trust anchors in your account but no service-linked role. In the AWS CLI or the AWS API, create a service-linked role with the `rolesanywhere.amazonaws.com` service name. For more information, see Creating a service-linked role in the *IAM User Guide*. If you delete this service-linked role, you can use this same process to create the role again.

Editing a service-linked role for IAM Roles Anywhere

IAM Roles Anywhere does not allow you to edit the AWSServiceRoleForRolesAnywhere service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

Deleting a service-linked role for IAM Roles Anywhere

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

> **Note**
>
> If IAM Roles Anywhere is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To delete IAM Roles Anywhere resources used by the AWSServiceRoleForRolesAnywhere

- Delete all trust anchors in your account in all Regions that contain them.

To manually delete the service-linked role using IAM

- For information about using the IAM console, the AWS CLI, or the AWS API to delete the AWSServiceRoleForRolesAnywhere service-linked role, see Deleting a service-linked role in the *IAM User Guide*.

Supported regions for IAM Roles Anywhere service-linked roles

IAM Roles Anywhere supports using service-linked roles in all of the regions where the service is available. For more information, see AWS regions and endpoints.

Region name	Region identity	Support in IAM Roles Anywhere
US East (N. Virginia)	us-east-1	Yes
US East (Ohio)	us-east-2	Yes
US West (N. California)	us-west-1	Yes
US West (Oregon)	us-west-2	Yes
Asia Pacific (Mumbai)	ap-south-1	Yes
Asia Pacific (Osaka)	ap-northeast-3	Yes
Asia Pacific (Seoul)	ap-northeast-2	Yes
Asia Pacific (Singapore)	ap-southeast-1	Yes
Asia Pacific (Sydney)	ap-southeast-2	Yes
Asia Pacific (Tokyo)	ap-northeast-1	Yes
Asia Pacific (Hong Kong)	ap-east-1	Yes
Asia Pacific (Jakarta)	ap-southeast-3	Yes
Canada (Central)	ca-central-1	Yes
Europe (Frankfurt)	eu-central-1	Yes
Europe (Ireland)	eu-west-1	Yes
Europe (London)	eu-west-2	Yes
Europe (Paris)	eu-west-3	Yes

Region name	Region identity	Support in IAM Roles Anywhere
Europe (Milan)	eu-south-1	Yes
Europe (Stockholm)	eu-north-1	Yes
Africa (Cape Town)	af-south-1	Yes
South America (São Paulo)	sa-east-1	Yes
Middle East (Bahrain)	me-south-1	Yes
Asia Pacific (Hyderabad)	ap-south-2	Yes
Europe (Zurich)	eu-central-2	Yes
Europe (Spain)	eu-south-2	Yes
Middle East (UAE)	me-central-1	Yes
Asia Pacific (Melbourne)	ap-southeast-4	Yes
Israel (Tel Aviv)	il-central-1	Yes
Canada West (Calgary)	ca-west-1	Yes
Asia Pacific (Thailand)	ap-southeast-7	Yes
Asia Pacific (Malaysia)	ap-southeast-5	Yes
Mexico (Central)	mx-central-1	Yes
Asia Pacific (Taipei)	ap-east-2	Yes
AWS GovCloud (US-West)	us-gov-west-1	Yes
AWS GovCloud (US-East)	us-gov-east-1	Yes

AWS managed policies for IAM Roles Anywhere

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AWSRolesAnywhereServicePolicy

AWSRolesAnywhereServicePolicy provides the required permissions to publish metrics data to CloudWatch namespaces (`AWS/RolesAnywhere` and `AWS/Usage`). This policy also grants permissions to list information about your private certificate authority (CA) and retrieve the certificate and certificate chain for your private CA from AWS Private CA.

You can't attach AWSRolesAnywhereServicePolicy to your IAM entities. This policy is attached to a service-linked role that allows IAM Roles Anywhere to perform actions on your behalf. For more information, see Service-linked role permissions for IAM Roles Anywhere.

This policy grants administrative permissions that allow IAM Roles Anywhere to publish CloudWatch metrics and check the configuration of AWS Private CA.

Permissions details

This policy includes the following permissions.

- `cloudwatch` – Allows principals to publish metric data points to the `AWS/RolesAnywhere` and `AWS/Usage` namespaces.

- acm-pca – Allows principals to list information about your private certificate authority (CA) and retrieve the certificate and certificate chain for your private CA from AWS Private CA.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "cloudwatch:PutMetricData"
        ],
        "Resource": "*",
        "Condition": {
          "StringEquals": {
            "cloudwatch:namespace": [
              "AWS/RolesAnywhere",
              "AWS/Usage"
            ]
          }
        }
      },
      {
        "Effect": "Allow",
        "Action": [
          "acm-pca:GetCertificateAuthorityCertificate",
          "acm-pca:DescribeCertificateAuthority"
        ],
        "Resource": "arn:aws:acm-pca:*:*:*"
      }
    ]
  }
```

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
```

```
        "Effect": "Allow",
        "Action": [
          "cloudwatch:PutMetricData"
        ],
        "Resource": "*",
        "Condition": {
          "StringEquals": {
            "cloudwatch:namespace": [
              "AWS/RolesAnywhere",
              "AWS/Usage"
            ]
          }
        }
      }
    ]
}
```

For more information and definition of this policy, see: AWSRolesAnywhereServicePolicy.

AWS managed policy: AWSRolesAnywhereFullAccess

This policy provides all permissions to IAM Roles Anywhere resources, including but not limited to: CreateProfile, DeleteTrustAnchor, DisableCRL, ResetNotificationSettings.

You can attach the AWSRolesAnywhereFullAccess policy to your IAM identities.

This policy grants full access permissions that allow users to view, create, update, and delete all IAM Roles Anywhere resources.

Permissions details

This policy includes the following permissions:

- `rolesanywhere` – Allows principals to perform all actions on IAM Roles Anywhere resources including trust anchors, profiles, CRLs, subjects, and notification settings.
- `iam:PassRole` – Allows principals to pass a role to IAM Roles Anywhere.
- `iam:CreateServiceLinkedRole` – Allows principals to create the service-linked role for IAM Roles Anywhere. This permission is needed when the service-linked role doesn't exist in the account yet.

For more information and definition of this policy, see AWSRolesAnywhereFullAccess.

AWS managed policy: AWSRolesAnywhereReadOnly

This policy provides read-only permissions to IAM Roles Anywhere resources, including but not limited to: GetTrustAnchor, ListProfiles, GetCRL.

You can attach the AWSRolesAnywhereReadOnly policy to your IAM identities.

This policy grants read-only permissions that allow users to view IAM Roles Anywhere resources but not modify them.

Permissions details

This policy includes the following permissions:

- `rolesanywhere` – Allows principals to view all IAM Roles Anywhere resources including trust anchors, profiles, CRLs, and subjects.

For more information and definition of this policy, see AWSRolesAnywhereReadOnly.

IAM Roles Anywhere updates to AWS managed policies

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

Change	Description	Date
AWSRolesAnywhereFullAccess – New policy	IAM Roles Anywhere added a new policy to allow users to grant full access IAM Roles Anywhere permissions to principals in a standardized way.	July 16, 2025
AWSRolesAnywhereReadOnly – New policy	IAM Roles Anywhere added a new policy to allow users	July 16, 2025

Change	Description	Date
	to grant read only IAM Roles Anywhere permissions to principals in a standardized way.	
IAM Roles Anywhere started tracking changes	IAM Roles Anywhere started tracking changes for its AWS managed policies.	February 27, 2023

Disaster recovery and resilience in IAM Roles Anywhere

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.

IAM Roles Anywhere and interface VPC endpoints (AWS PrivateLink)

You can establish a private connection between your VPC and AWS Identity and Access Management Roles Anywhere by creating an *interface VPC endpoint*. Interface endpoints are powered by AWS PrivateLink, a technology that enables you to privately access IAM Roles Anywhere APIs without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to communicate with IAM Roles Anywhere APIs. Traffic between your VPC and IAM Roles Anywhere does not leave the Amazon network.

Each interface endpoint is represented by one or more Elastic Network Interfaces in your subnets.

For more information, see Interface VPC endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*.

Considerations for IAM Roles Anywhere VPC endpoints

Before you set up an interface VPC endpoint for IAM Roles Anywhere, ensure that you review Interface endpoint properties and limitations in the *Amazon VPC User Guide*.

IAM Roles Anywhere supports making calls to all of its API actions from your VPC.

> **Note**
>
> VPC endpoint policies are supported for IAM Roles Anywhere on all API methods except `CreateSession`.

Full access to IAM Roles Anywhere is allowed through the endpoint by default, including `CreateSession`. For more information, see Controlling access to services with VPC endpoints in the *Amazon VPC User Guide*.

Creating an interface VPC endpoint for IAM Roles Anywhere

You can create a VPC endpoint for the IAM Roles Anywhere service using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Creating an interface endpoint in the *Amazon VPC User Guide*.

Create a VPC endpoint for IAM Roles Anywhere using the following service name:

- com.amazonaws.*region*.rolesanywhere

If you enable private DNS for the endpoint, you can make API requests to IAM Roles Anywhere using its default DNS name for the Region, for example, `rolesanywhere.us-east-1.amazonaws.com`.

For more information, see Accessing a service through an interface endpoint in the *Amazon VPC User Guide*.

Creating a VPC endpoint policy for IAM Roles Anywhere

You can attach an endpoint policy to your VPC endpoint that controls access to IAM Roles Anywhere. The policy specifies the following information:

- The principal that can perform actions.

- The actions that can be performed.
- The resources on which actions can be performed.

For more information, see Controlling access to services with VPC endpoints in the *Amazon VPC User Guide*.

Example: VPC endpoint policy for IAM Roles Anywhere actions

The following is an example of an endpoint policy for IAM Roles Anywhere. When attached to an endpoint, this policy grants access to the listed IAM Roles Anywhere actions for all principals on all resources.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "rolesanywhere:CreateTrustAnchor",
            "rolesanywhere:CreateProfile",
            "rolesanywhere:UpdateProfile"
         ],
         "Resource":"*"
      }
   ]
}
```

Control API access with IAM policies

If you use IAM policies to control access to AWS services based on IP addresses, you might need to update your policies to include IPv6 address ranges. This guide explains the differences between IPv4 and IPv6 and describes how to update your IAM policies to support both protocols. Implementing these changes helps you maintain secure access to your AWS resources while supporting IPv6.

What is IPv6?

IPv6 is the next generation IP standard intended to eventually replace IPv4. The previous version, IPv4, uses a 32-bit addressing scheme to support 4.3 billion devices. IPv6 instead uses 128-bit addressing to support approximately 340 trillion trillion trillion (or 2 to the 128th power) devices.

For more information, see the VPC IPv6 web page.

These are examples of IPv6 addresses:

```
2001:cdba:0000:0000:0000:0000:3257:9652 # This is a full, unabbreviated IPv6 address.
2001:cdba:0:0:0:0:3257:9652             # The same address with leading zeros in each
 group omitted
2001:cdba::3257:965                     # A compressed version of the same address.
```

IAM dual-stack (IPv4 and IPv6) policies

You can use IAM policies to control access to IAM Roles Anywhere APIs and prevent IP addresses outside the configured range from accessing IAM Roles Anywhere APIs.

The *rolesanywhere.{region}.api.aws* dual-stack endpoint for IAM Roles Anywhere APIs supports both IPv6 and IPv4.

If you need to support both IPv4 and IPv6, update your IP address filtering policies to handle IPv6 addresses. Otherwise, you might not be able to connect to IAM Roles Anywhere over IPv6.

Who should make this change?

This change affects you if you use dual addressing with policies that contain `aws:sourceIp`. *Dual addressing* means that the network supports both IPv4 and IPv6.

If you use dual addressing, update your IAM policies that currently use IPv4 format addresses to include IPv6 format addresses.

Who should not make this change?

This change doesn't affect you if you *only* use IPv4 networks.

Adding IPv6 to an IAM policy

IAM policies use the `aws:SourceIp` condition key to control access from specific IP addresses.If your network uses dual addressing (IPv4 and IPv6), update your IAM policies to include IPv6 address ranges.

In the `Condition` element of your policies, use the `IpAddress` and `NotIpAddress` operators for IP address conditions. Don't use string operators, as they can't handle the various valid IPv6 address formats.

These examples use `aws:SourceIp`. For VPCs, use `aws:VpcSourceIp` instead.

The following is the Denies access to AWS based on the source IP reference policy from the *IAM User Guide*. The `NotIPAddress` in the `Condition` element to lists two IPv4 address ranges, `192.0.2.0/24` and `203.0.113.0/24`, which will be denied access to the API.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*",
        "Condition": {
            "NotIpAddress": {
                "aws:SourceIp": [
                    "192.0.2.0/24",
                    "203.0.113.0/24"
                ]
            },
            "Bool": {
                "aws:ViaAWSService": "false"
            }
        }
    }
}
```

To update this policy, change the `Condition` element to include the IPv6 address ranges `2001:DB8:1234:5678::/64` and `2001:cdba:3257:8593::/64`.

> **Note**
>
> Don't remove the existing IPv4 addresses. They're needed for backward compatibility.

```
"Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24", <<DO NOT REMOVE existing IPv4 address>>
```

```
                    "203.0.113.0/24", <<DO NOT REMOVE existing IPv4 address>>
                    "2001:DB8:1234:5678::/64", <<New IPv6 IP address>>
                    "2001:cdba:3257:8593::/64" <<New IPv6 IP address>>
                ]
            },
            "Bool": {
                "aws:ViaAWSService": "false"
            }
        }
```

Verifying your client supports IPv6

If you use the *rolesanywhere.{region}.api.aws* endpoint, verify that you can connect to it. The following steps describe how to perform the verification.

This examples uses Linux and curl version 8.6.0 and uses the [AWS Identity and Access Management Roles Anywhere service endpoints] which has IPv6 enabled endpoints located at the **api.aws** endpoint.

> **Note**
>
> Switch the AWS Region to the same Region where your service is located. In this example, we use the US East (N. Virginia) – `us-east-1` endpoint.

1. Determine if the endpoint resolves with an IPv6 address using the following `dig` command.

```
$ dig +short AAAA rolesanywhere.us-east-1.api.aws

> 2600:1f18:e2f:4e05:1a8a:948e:7c08:c1c3
```

2. Determine if the client network can make an IPv6 connection using the following `curl` command. A 404 response code means the connection succeeded, while a 0 response code means the connection failed.

```
$ curl --ipv6 -o /dev/null --silent -w "\nremote ip: %{remote_ip}\nresponse code:
 %{response_code}\n" https://rolesanywhere.us-east-1.api.aws

> remote ip: 2600:1f18:e2f:4e05:1a8a:948e:7c08:c1c3
> response code: 404
```

If a remote IP was identified **and** the response code is not `0`, a network connection was successfully made to the endpoint using IPv6. The remote IP should be an IPv6 address because the operating system should select the protocol that is valid for the client. If the remote IP is not an IPv6 address, use the following command to force `curl` to use IPv4.

```
$ curl --ipv4 -o /dev/null --silent -w "\nremote ip: %{remote_ip}\nresponse code:
 %{response_code}\n" https://rolesanywhere.us-east-1.api.aws

> remote ip: 3.123.154.250
> response code: 404
```

If the remote IP is blank or the response code is `0`, the client network or the network path to the endpoint is IPv4-only. You can verify this configuration with the following `curl` command.

```
$ curl -o /dev/null --silent -w "\nremote ip: %{remote_ip}\nresponse code:
 %{response_code}\n" https://rolesanywhere.us-east-1.api.aws

> remote ip: 3.123.154.250
> response code: 404
```

AWS services for monitoring IAM Roles Anywhere

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

- *Amazon CloudWatch* monitors your AWS resources and the applications you run on AWS in real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For example, you can have CloudWatch track CPU usage or other metrics of your Amazon EC2 instances and automatically launch new instances when needed. For more information, see the Amazon CloudWatch User Guide.
- *Amazon EventBridge* can be used to automate your AWS services and respond automatically to system events, such as application availability issues or resource changes. Events from AWS services are delivered to EventBridge in near real time. You can write simple rules to indicate which events are of interest to you and which automated actions to take when an event matches a rule. For more information, see Amazon EventBridge User Guide.
- *Amazon EventBridge* is a serverless event bus service that makes it easy to connect your applications with data from a variety of sources. EventBridge delivers a stream of real-time data from your own applications, Software-as-a-Service (SaaS) applications, and AWS services and routes that data to targets such as Lambda. This enables you to monitor events that happen in services, and build event-driven architectures. For more information, see the Amazon EventBridge User Guide.
- *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.

Customize notification settings in IAM Roles Anywhere

You can customize notification settings based on your public key infrastructure. These settings are attached to your trust anchor and allow you to define custom thresholds for a notification event. IAM Roles Anywhere will consume these settings while evaluating for a notification event to send metrics/events/notifications through their respective notification channels.

Topics

- Notification events
- Notification channels
- IAM Roles Anywhere default notification settings
- Notification evaluation criteria
- Configuring custom notification threshold (console)
- Disabling a notification setting (console)

Notification events

- **CA certificate expiry**: IAM Roles Anywhere sends notification when a certificate authority (CA) in your trust anchor is approaching expiry.
- **End-entity certificate expiry**: IAM Roles Anywhere sends notification when your end-entity certificate used to vend temporary security credentials is expiring soon.

Notification channels

> **Note**
>
> Notification channel with a value of ALL will apply the custom settings to all the channels listed below.

- Amazon CloudWatch metrics
- Amazon EventBridge events
- AWS Health notifications

IAM Roles Anywhere default notification settings

Following are the default notification settings IAM Roles Anywhere has defined. These values are applied in the absence of custom notification settings.

Event	Channel	Threshold	Enabled
CA certificate expiry	CloudWatch, EventBridge and AWS Health	45 days before expiry	True
End entity certificate expiry	EventBridge and AWS Health	45 days before expiry	True

Notification evaluation criteria

Following are the evaluation criteria used to send notification events.

These criteria do not apply if your notification setting is in a `disabled` state.

Event	Channel	Starts when	Ends at
CA certificate expiry	CloudWatch	Number of days until certificate expiry ≤ threshold	Day of certificate expiry
CA certificate expiry	EventBridge and AWS Health	Number of days until certificate expiry ≤ threshold	14 days after certificate expires
End-entity certificate expiry	EventBridge and AWS Health	Number of days until certificate expiry ≤ threshold	Day of certificate expiry

Configuring custom notification threshold (console)

1. Sign in to IAM Roles Anywhere console.
2. Scroll to find trust anchor table and **choose the trust anchor** to apply custom notification settings.
3. Within trust anchor detail page scroll towards **Notification settings** section and choose **Manage settings**.

4. **Customize threshold** for the notification event. IAM Roles Anywhere will start sending metrics/events/notifications when number of days until your X.509 certificate expires is less than or equal this threshold. See IAM Roles Anywhere notification evaluation criteria.
5. Choose **Save changes** to apply custom notification threshold.

Disabling a notification setting (console)

1. Sign in to IAM Roles Anywhere console.
2. Scroll to find trust anchor table and **choose the trust anchor** to apply custom notification settings.
3. Within trust anchor detail page scroll towards **Notification settings** section and choose **Manage settings**.
4. **Choose the table cell** from `Status` column for notification event name **End entity certificate expiry**.
5. From the options displayed in the selection pane choose the **Disable** option.
6. Choose **Save changes** to apply to disable notification settings for end-entity certificate expiry event.

Monitoring IAM Roles Anywhere with Amazon CloudWatch

You can monitor AWS Identity and Access Management Roles Anywhere using CloudWatch, which collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, see the Amazon CloudWatch User Guide.

For IAM Roles Anywhere, you might want to watch for trust anchor and end-entity certificates expiration dates and renew your certificates when your certificates are nearing expiration.

The IAM Roles Anywhere service reports the following metrics in the `AWS/RolesAnywhere` namespace.

Metric	Description
`Success`	Gets published every time `CreateSession` succeeds in returning credentials to the user. Valid Dimensions: Operation, TrustAnchorArn Valid Statistic: Sum Units: Count
`Failure`	Gets published every time `CreateSession` fails to return credentials to the user. Valid Dimensions: Operation, ErrorType Valid Statistic: Sum Units: Count
`DaysToExpiry`	Gets published every time trust anchor certificates satisfies notification evaluation criteria. This metric will be published at most once a day. Valid Dimensions: TrustAnchorArn Units: Integer

The following dimensions are supported for the IAM Roles Anywhere metrics.

Dimension	Description
`Operation`	The operation for which the metric applies to. This can only take on the value, CreateSession.
`TrustAnchorArn`	The ARN of the trust anchor that is relevant for this metric.
`ErrorType`	The type of error that `CreateSession` errors out with.

Monitoring IAM Roles Anywhere events in Amazon EventBridge

You can monitor IAM Roles Anywhere events in Amazon EventBridge. Events from IAM Roles Anywhere are delivered to EventBridge in near-real time. You can write simple rules to indicate which events are of interest to you and the automated actions to take when an event matches a rule. With EventBridge, you can use events to trigger targets including AWS Lambda functions, AWS Batch jobs, Amazon SNS topics, and many others. For more information, see Creating Amazon EventBridge rules that react to events.

The following examples show events for IAM Roles Anywhere.

Topics

- Trust anchor certificate expiration event
- Intermediate or end-entity certificate expiration event
- Responding to an event

Trust anchor certificate expiration event

IAM Roles Anywhere sends daily expiration event for each trust anchor certificate that satisfies notification evaluation criteria. You can use expiration events to configure Amazon SNS to send a text notification whenever IAM Roles Anywhere generates this event.

Expiration events have the following structure.

```
{
  "version": "0",
  "id": "9c95e8e4-96a4-ef3f-b739-b6aa5b193afb",
  "detail-type": "Roles Anywhere Certificate Expiration State Change",
  "source": "aws.rolesanywhere",
  "account": "123456789012",
  "time": "2022-06-10T06:51:08Z",
  "region": "us-west-1",
  "resources": [
    "arn:aws:rolesanywhere:us-west-1:123456789012:trust-anchor/61f50cd4-45b9-4259-b049-
d0a53682fa4b"
  ],
  "detail": {
    "certificate-serial-number": "00936EACBE07F201DF",
    "days-to-expiry": 3,
```

```
    "issuer": "L=Seattle,CN=CA Root v1,ST=Washington,C=US"
  }
}
```

Intermediate or end-entity certificate expiration event

IAM Roles Anywhere sends an expiration event for intermediate or end-entity certificates when the certificate satisfies notification evaluation criteria and used in createSession API. You can use expiration events to configure Amazon SNS to send a text notification whenever IAM Roles Anywhere generates this event.

Expiration events have the following structure.

```
{
  "version": "0",
  "id": "9c95e8e4-96a4-ef3f-b739-b6aa5b193afb",
  "detail-type": "Roles Anywhere Certificate Expiration State Change",
  "source": "aws.rolesanywhere",
  "account": "123456789012",
  "time": "2022-06-10T06:51:08Z",
  "region": "us-west-1",
  "detail": {
    "certificate-serial-number": "00936EACBE07F201DF",
    "days-to-expiry": 3,
    "issuer": "L=Seattle,CN=CA Root v1,ST=Washington,C=US"
  }
}
```

Responding to an event

You can configure Amazon Simple Notification Service to send a text notification whenever IAM Roles Anywhere generates an EventBridge event.

To create an Amazon EventBridge rule that reacts to events

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose **Rules**.
3. Choose **Create rule**.
4. Enter a name and description for the rule.

A rule can't have the same name as another rule in the same Region and on the same event bus.

5. For **Event bus**, choose the event bus that you want to associate with this rule. If you want this rule to match events that come from your account, select **AWS default event bus**. When an AWS service in your account emits an event, it always goes to your account's default event bus.
6. For **Rule type**, choose **Rule with an event pattern**.
7. Choose **Next**.
8. For **Event source**, choose **AWS services**.
9. For **Sample events**, choose an event under **IAM Roles Anywhere**.
10. For **Event pattern**, do the following:

    a. For **Event source**, choose **AWS services**.

    b. For **AWS service**, choose **IAM Roles Anywhere.**

    c. For **Event Type**, choose an **IAM Roles Anywhere** event.

    d. Choose **Next**
11. In the **Targets** section, choose a service that can consume your event such as Amazon SNS, or choose **Lambda function** to pass the event to customized executable code.

Monitoring IAM Roles Anywhere notifications with AWS Health

You can monitor IAM Roles Anywhere health notifications in AWS Health. Notifications from IAM Roles Anywhere are delivered to AWS Health when certificates (both CA certificates in trust anchors and end-entity certificates) that are configured with IAM Roles Anywhere are nearing expiry. You can use these AWS Health notifications to take renewal actions on your certificates. For more information see Monitoring AWS Health events with Amazon EventBridge

Affected resources for trust anchor expiry notifications

IAM Roles Anywhere sends daily expiry notifications for each trust anchor that satisfies the notification evaluation criteria. For these notifications, the "Affected Resources" will each be trust anchors. If you have multiple certificates within a single trust anchor, it's possible that multiple are nearing expiry. IAM Roles Anywhere will determine whether a notification should be sent for a given trust anchor based on the certificate in the trust anchor that is expiring the soonest. Thus, you'll have to check each certificate in the trust anchor and take the necessary actions so as

to not cause impact to your workloads that rely on IAM Roles Anywhere for temporary security credentials.

Affected resources for end-entity certificate expiry notifications

IAM Roles Anywhere also sends daily expiry notifications for each end-entity certificate that was used to authenticate over the last day and satisfies the notification evaluation criteria. For these notifications, the "Affected Resources" will each be end-entity certificates. Each of these end-entity certificates will have a composite "Resource ID/ARN", of the form given below.

```
serialNumber=SerialNumber;certificateId=CertificateId
```

The `serialNumber` in the above resource identifier will contain the value of the serial number of the end-entity certificate that was used for authentication and will be expiring soon. And the `certificateId` in the above resource identifier will contain the value of the certificate ID for that certificate. The certificate ID is defined as `Hex(SHA256(`*`ASN.1 DER Certificate Bytes`*`))`, where the result is a lowercase hex-encoded string. If you have a PEM file that contains your certificate data, you can use OpenSSL to convert your certificate into its DER representation and then take the SHA256 hash of the resulting value.

```
openssl x509 -in end-entity-certificate.pem -inform PEM -outform DER | sha256sum
```

Logging IAM Roles Anywhere API calls using AWS CloudTrail

AWS Identity and Access Management Roles Anywhere is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in IAM Roles Anywhere. CloudTrail captures all API calls for IAM Roles Anywhere as events. The calls captured include calls from the IAM Roles Anywhere console and code calls to the IAM Roles Anywhere API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for IAM Roles Anywhere. 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 IAM Roles Anywhere, 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.

IAM Roles Anywhere information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in IAM Roles Anywhere, 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 IAM Roles Anywhere, 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 IAM Roles Anywhere actions are logged by CloudTrail and are documented in the IAM Roles Anywhere API Reference. For example, calls to the `CreateTrustAnchor`, `ListProfiles`, and `CreateSession` operations generate entries in the CloudTrail log files. In addition, the userIdentity element's Role Session Name property is the hex-encoded serial number of the certificate the session was created with and can be used to track a session back to it.

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 IAM Roles Anywhere 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 `UpdateProfile` operation.


```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAZR5EMTJKE753U4ZDS:test-session",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/test-session",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAZR5EMTJKE753U4ZDS",
        "arn": "arn:aws:iam::111122223333:role/Admin",
        "accountId": "111122223333",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2022-03-21T22:40:46Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2022-07-01T18:11:27Z",
  "eventSource": "rolesanywhere.amazonaws.com",
  "eventName": "UpdateProfile",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "1.1.1.1",
  "userAgent": "test-agent",
  "requestParameters": {
    "durationSeconds": 3600,
```

```
    "managedPolicyArns": [
        "arn:aws:iam::aws:policy/AdministratorAccess",
        "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
    ],
    "name": "Updated Test Profile",
    "profileId": "0ace5b12-24b9-427e-a483-c55884852fbf",
    "sessionPolicy": "{\n  \"Version\":\"2012-10-17\",\n  \"Statement\":[\n    {\n
   \"Effect\":\"Allow\",\n      \"Action\":\"s3:ListObjects\",\n      \"Resource\":\"*
\"\n    }\n  ]\n}\n"
},
"responseElements": {
  "profile": {
      "createdAt": "2022-07-01T18:11:27.380711Z",
      "createdBy": "arn:aws:sts::111122223333:assumed-role/Admin/test-session",
      "durationSeconds": 3600,
      "enabled": false,
      "managedPolicyArns": [
          "arn:aws:iam::aws:policy/AdministratorAccess",
          "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
      ],
      "name": "Updated Test Profile",
      "profileArn": "arn:aws:rolesanywhere:us-
east-1:111122223333:profile/0ace5b12-24b9-427e-a483-c55884852fbf",
      "profileId": "0ace5b12-24b9-427e-a483-c55884852fbf",
      "requireInstanceProperties": false,
      "roleArns": [
          "arn:aws:iam::111122223333:role/test-role"
      ],
      "sessionPolicy": "{\n  \"Version\":\"2012-10-17\",\n  \"Statement\":[\n    {\n
    \"Effect\":\"Allow\",\n      \"Action\":\"s3:ListObjects\",\n      \"Resource\":\"*
\"\n    }\n  ]\n}\n",
      "updatedAt": "2022-07-01T18:11:27.936687Z"
  }
},
  "requestID": "ca28860f-504a-4f2d-9f3f-f9cfb4ba0491",
  "eventID": "a7bb90c3-c47b-4832-88e7-aeaccda21f1a",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "eventCategory": "Management",
  "tlsDetails": {
    "clientProvidedHostHeader": "rolesanywhere.us-east-1.amazonaws.com"
  }
```

```
}
```

Monitoring authentications with IAM Roles Anywhere subjects

You can use the **Subject Activity** tab in the IAM Roles Anywhere console to visualize and audit activities for certificates that are authenticated with IAM Roles Anywhere. A *subject* represents a unique identity defined by the X.509 subject of any certificates you use to authenticate with IAM Roles Anywhere. IAM Roles Anywhere creates a subject for you at the time of authentication if there isn't one already for the X.509 subject. Each subject contains the most recent certificates you have used with IAM Roles Anywhere.

To view the history of an X.509 subject

1. Sign in to the IAM Roles Anywhere console.
2. Navigate to the **Subject activity** tab.
3. In the list of certificates records grouped by X.509 **Subject**, choose the **Subject** record that you want to check.
4. On the **Subject details** page, view the details of the subject record.
5. In the **Certificates** section, you can see the most recent record for certificates authenticated with IAM Roles Anywhere that have the same certificate subject.
6. Choose the **Serial number** record to view or copy the certificate body.

Create IAM Roles Anywhere resources with AWS CloudFormation

AWS Identity and Access Management Roles Anywhere is integrated with AWS CloudFormation, a service that helps you to model and set up your AWS resources so that you can spend less time creating and managing your resources and infrastructure. You create a template that describes all the AWS resources that you want (such as `AWS::RolesAnywhere::Crl`, `AWS::RolesAnywhere::Profile`, and `AWS::RolesAnywhere::TrustAnchor`), and AWS CloudFormation provisions and configures those resources for you. For more information, see resource type reference in the AWS CloudFormation User Guide.

When you use AWS CloudFormation, you can reuse your template to set up your IAM Roles Anywhere resources consistently and repeatedly. Describe your resources once, and then provision the same resources over and over in multiple AWS accounts and Regions.

IAM Roles Anywhere and AWS CloudFormation templates

To provision and configure resources for IAM Roles Anywhere and related services, you must understand AWS CloudFormation templates. Templates are formatted text files in JSON or YAML. These templates describe the resources that you want to provision in your AWS CloudFormation stacks. If you're unfamiliar with JSON or YAML, you can use AWS CloudFormation Designer to help you get started with AWS CloudFormation templates. For more information, see What is AWS CloudFormation Designer? in the *AWS CloudFormation User Guide*.

IAM Roles Anywhere supports creating certificate revocation lists, trust anchors, and profiles in AWS CloudFormation. For more information, including examples of JSON and YAML templates for CRL, TrustAnchor, and Profile, see the *AWS CloudFormation User Guide*.

Learn more about AWS CloudFormation

To learn more about AWS CloudFormation, see the following resources:

- AWS CloudFormation
- AWS CloudFormation User Guide
- AWS CloudFormation API Reference

- AWS CloudFormation Command Line Interface User Guide

Quotas for AWS Identity and Access Management Roles Anywhere

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, and other quotas cannot be increased.

To view the quotas for AWS Identity and Access Management Roles Anywhere, open the Service Quotas console. In the navigation pane, choose **AWS services** and select **IAM Roles Anywhere**.

To request a quota increase, see Requesting a Quota Increase in the *Service Quotas User Guide*. If the quota is not yet available in Service Quotas, use the limit increase form.

Your AWS account has the following quotas related to IAM Roles Anywhere and each quota is per AWS Region.

Resource	Description	Default value	Adjustable
Combined rate of trust anchor requests	The maximum transactions per second for ListTrustAnchors, CreateTrustAnchor, GetTrustAnchor, UpdateTrustAnchor, DeleteTrustAnchor, EnableTrustAnchor, and DisableTrustAnchor requests combined.	1 per second	Yes
Combined rate of profile requests	The maximum transactions per second for ListProfiles, CreateProfile, GetProfile, UpdateProfile, DeleteProfile,	1 per second	Yes

Resource	Description	Default value	Adjustable
	EnableProfile, and DisableProfile requests combined.		
Combined rate of subject requests	The maximum transactions per second for ListSubjects and GetSubject requests combined.	1 per second	Yes
Combined rate of tagging requests	The maximum transactions per second for TagResource, UntagResource, and ListTagsForResource requests combined.	1 per second	Yes
Combined rate of CRL requests	The maximum transactions per second for ListCrls, GetCrl, ImportCrl, UpdateCrl, DeleteCrl, EnableCrl, and DisableCrl requests combined.	1 per second	Yes
Rate of CreateSession requests	The maximum transactions per second for CreateSession requests.	10 per second	Yes

Resource	Description	Default value	Adjustable
Trust anchors	The maximum number of trust anchors that you can create within an account.	50	Yes
Profiles	The maximum number of profiles that you can create within an account.	250	Yes
CRLs per trust anchor	The maximum number of Certificate Revocation Lists (CRLs) that you can create per trust anchor within an account.	2	No
Certificates per trust anchor	The maximum number of certificates that you can create per trust anchor within an account.	2	No
Roles per profile	The maximum number of roles that you can create per profile within an account.	250	No

Throttling

Workloads obtain session credentials by using an endpoint that does not use AWS authenticated principals, which would typically be used to limit the rate of operations. IAM Roles Anywhere will limit the rate of calls to the credential endpoint by the authenticating certificate information and IP address (including VPC Endpoint, if applicable).

Document history for the AWS Identity and Access Management Roles Anywhere User Guide

The following table describes the documentation releases for IAM Roles Anywhere.

Change	Description	Date
IAM Roles Anywhere released new AWS managed policies	Introduced new policies: AWSRolesAnywhereReadOnly and AWSRolesAnywhereFullAccess.	July 16, 2025
Released Credential Helper version 1.7.0	IAM Roles Anywhere released Credential Helper version 1.7.0. For more information, see Credential Helper Changelog.	June 2, 2025
Added IPv6 and dual-stack endpoint	Added section for managing access to IPv4 and IPv6 endpoints. See https://docs.aws.amazon.com/rolesanywhere/latest/userguide/ip-access.html	December 18, 2024
Updated the service-linked role description	Updated the service-linked role description and added role permissions policy. For more information, see Service-linked role permissions for IAM Roles Anywhere.	February 27, 2023
Released Credential Helper version 1.0.4	IAM Roles Anywhere released Credential Helper version 1.0.4. For more informati	January 17, 2023

	on, see Credential Helper Changelog.	
Released Credential Helper version 1.0.3	IAM Roles Anywhere released Credential Helper version 1.0.3. For more informati on, see Credential Helper Changelog.	December 5, 2022
Released Credential Helper version 1.0.2	IAM Roles Anywhere released Credential Helper version 1.0.2. For more informati on, see Credential Helper Changelog.	September 8, 2022
Added Signing Process and CreateSession	Added Signing Process and CreateSession sections. For more information, see Signing process and CreateSession	July 15, 2022
Released Credential Helper version 1.0.1	IAM Roles Anywhere released Credential Helper version 1.0.1. For more informati on, see Credential Helper Changelog.	July 14, 2022
Added certificate revocation	Added certificate revocatio n in the IAM Roles Anywhere trust model page. For more information, see Revocation.	July 13, 2022
Initial release	Initial release of the IAM Roles Anywhere User Guide	July 5, 2022