



User Guide

AWS Verified Access



AWS Verified Access: User Guide

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is AWS Verified Access?	1
Benefits of Verified Access	1
Accessing Verified Access	1
Pricing	2
How Verified Access works	3
Key components of Verified Access	3
Get started tutorial	5
Prerequisites	5
Create a trust provider	6
Create an instance	6
Create a group	7
Create an endpoint	7
Configure DNS for the endpoint	8
Test connectivity to the application	9
Add an access policy	9
Clean up	9
Verified Access instances	11
Create and manage a Verified Access instance	11
Create a Verified Access instance	11
Attach a trust provider to a Verified Access instance	12
Detach a trust provider from a Verified Access instance	12
Add a custom subdomain	13
Delete a Verified Access instance	13
Integrate with AWS WAF	14
Required IAM permissions	15
Associate an AWS WAF web ACL	15
Check the status of the association	16
Disassociate an AWS WAF web ACL	16
FIPS compliance	17
Existing environment	17
New environment	18
Trust providers	19
User-identity	19
IAM Identity Center	19

OIDC trust provider	21
Device-based	24
Supported device trust providers	24
Create a device-based trust provider	25
Modify a device-based trust provider	25
Delete a device-based trust provider	26
Verified Access groups	27
Create and manage a Verified Access group	27
Create a Verified Access group	28
Modify a Verified Access group	28
Modify a Verified Access group policy	29
Share a group with another account	29
Considerations	30
Resource shares	31
Delete a Verified Access group	31
Verified Access endpoints	33
Verified Access endpoint types	33
How Verified Access works with shared VPCs and subnets	34
Create a load balancer endpoint	34
Create a network interface endpoint	36
Create a network CIDR endpoint	37
Create an Amazon Relational Database Service endpoint	38
Allow traffic from your endpoint	40
Modify a Verified Access endpoint	41
Modify a Verified Access endpoint policy	41
Delete a Verified Access endpoint	42
Verified Access trust data	43
Default context	43
HTTP request	44
TCP flow	45
AWS IAM Identity Center context	46
Third-party context	48
Browser extension	48
Jamf	49
CrowdStrike	51
JumpCloud	53

User claims passing	54
JWT for OIDC user claims	55
JWT for IAM Identity Center user claims	56
Public keys	57
Retrieving and decoding JWT	57
Verified Access policies	59
Policy statements	59
Policy components	60
Comments	60
Multiple clauses	61
Reserved characters	61
Built-in operators	61
Policy evaluation	63
Policy logic short circuit	64
Example policies	65
Grant access to a group in IAM Identity Center	65
Grant access to a group in a third-party provider	66
Grant access using CrowdStrike	66
Allow or deny a specific IP address	66
Policy assistant	67
Step 1: Specify your resources	67
Step 2: Test and edit policies	68
Step 3: Review and apply changes	68
Connectivity Client	69
Prerequisites	69
Download the Connectivity Client	70
Export the client configuration file	70
Connect to the application	70
Uninstall the client	71
Best practices	71
Troubleshooting	72
When signing in, the browser doesn't open to complete authentication by the IdP	72
After authentication, the client status is "not connected"	72
Can't connect using a Chrome or Edge browser	73
Version history	73
Security	74

Data protection	74
Encryption in transit	75
Inter-network traffic privacy	76
Data encryption at rest	76
Identity and access management	90
Audience	91
Authenticating with identities	91
Managing access using policies	95
How Verified Access works with IAM	97
Identity-based policy examples	103
Troubleshooting	107
Use service-linked roles	108
AWS managed policies	110
Compliance validation	112
Resilience	113
Multiple subnets for high availability	113
Monitoring	114
Verified Access logs	114
Logging versions	115
Logging permissions	115
Enable or disable logs	116
Enable or disable trust context	118
OCSF version 0.1 log examples	119
OCSF version 1.0.0-rc.2 log examples	131
CloudTrail logs	138
Management events	140
Event examples	140
Quotas	142
Document history	144

What is AWS Verified Access?

With AWS Verified Access, you can provide secure access to your applications without requiring the use of a virtual private network (VPN). Verified Access evaluates each application request and helps ensure that users can access each application only when they meet the specified security requirements.

Benefits of Verified Access

- **Improved security posture** – A traditional security model evaluates access once and grants the user access to all applications. Verified Access evaluates each application access request in real time. This makes it difficult for bad actors to move from one application to another.
- **Integration with security services** – Verified Access integrates with identity and device management services, including both AWS and third-party services. Using data from these services, Verified Access verifies the trustworthiness of users and devices against a set of security requirements and determines whether the user should have access to an application.
- **Improved user experience** – Verified Access removes the need for users to use a VPN to access your applications. This helps reduce the number of support cases arising from VPN-related issues.
- **Simplified troubleshooting and audits** – Verified Access logs all access attempts, providing centralized visibility into application access, to help you quickly respond to security incidents and audit requests.

Accessing Verified Access

You can use any of the following interfaces to work with Verified Access:

- **AWS Management Console** – Provides a web interface that you can use to create and manage Verified Access resources. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
- **AWS Command Line Interface (AWS CLI)** – Provides commands for a broad set of AWS services, including AWS Verified Access. The AWS CLI is supported on Windows, macOS, and Linux. To get the AWS CLI, see [AWS Command Line Interface](#).

- **AWS SDKs** – Provide language-specific APIs. The AWS SDKs take care of many of the connection details, such as calculating signatures, and handling request retries and errors. For more information, see [AWS SDKs](#).
- **Query API** – Provides low-level API actions that you call using HTTPS requests. Using the Query API is the most direct way to access Verified Access. However, it requires your application to handle low-level details such as generating the hash to sign the request and handling errors. For more information, see [Verified Access actions](#) in the *Amazon EC2 API Reference*.

This guide describes how to use the AWS Management Console to create, access, and manage Verified Access resources.

Pricing

You are charged hourly for each application on Verified Access, and you are charged for the amount of data processed by Verified Access. For more information, see [AWS Verified Access pricing](#).

How Verified Access works

AWS Verified Access evaluates each application request from your users and allows access based on:

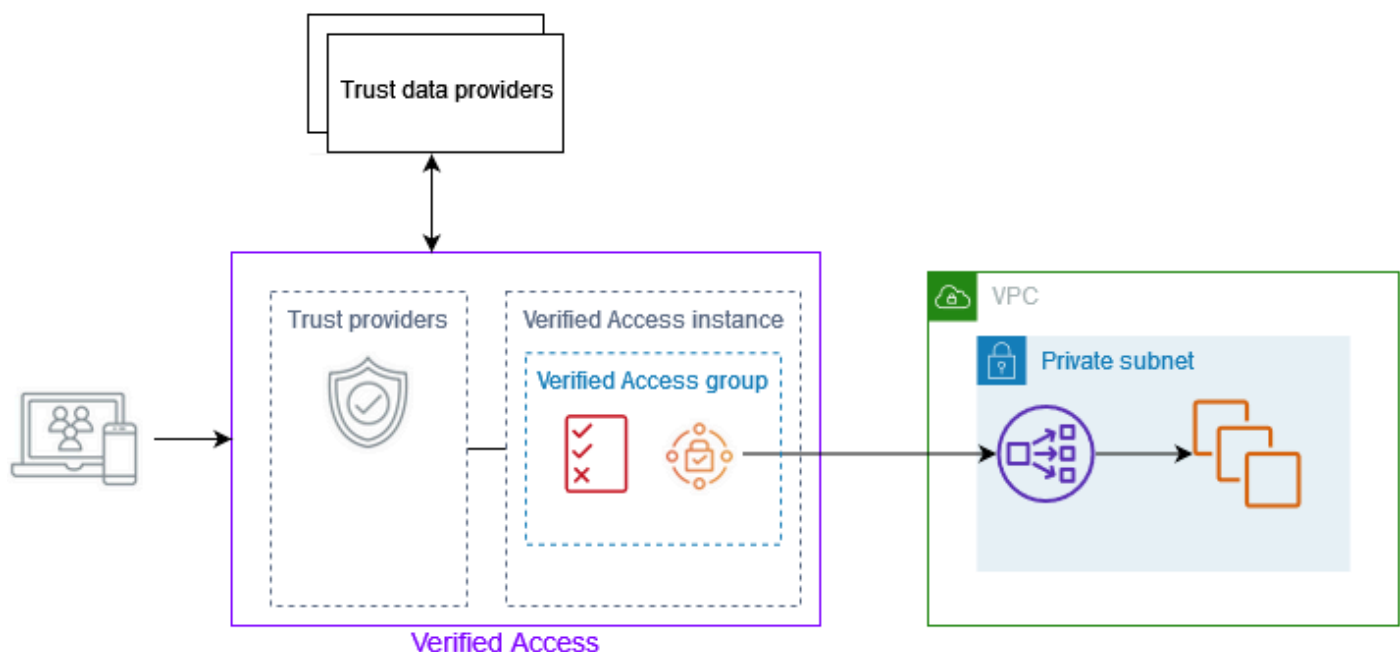
- Trust data sent by your chosen trust provider (from AWS or a third party).
- Access policies that you create in Verified Access.

When a user tries to access an application, Verified Access gets their data from the trust provider and evaluates it against the policies that you set for the application. Verified Access grants access to the requested application only if the user meets your specified security requirements. All application requests are denied by default, until a policy is defined.

In addition, Verified Access logs every access attempt, to help you respond quickly to security incidents and audit requests.

Key components of Verified Access

The following diagram provides a high-level overview of Verified Access. Users send requests to access an application. Verified Access evaluates the request against the access policy for the group and any application-specific endpoint policies. If access is allowed, the request is sent to the application through the endpoint.



- **Verified Access instances** – An instance evaluates application requests and grants access only when your security requirements are met.
- **Verified Access endpoints** – Each endpoint represents an application. In the diagram above, the application is hosted on EC2 instances that are targets of a load balancer.
- **Verified Access group** – A collection of Verified Access endpoints. We recommend that you group the endpoints for applications with similar security requirements to simplify policy administration. For example, you can group the endpoints for all your sales applications together.
- **Access policies** – A set of user-defined rules that determine whether to allow or deny access to an application. You can specify a combination of factors, including user identity and device security state. You create a group access policy for each Verified Access group, which is inherited by all endpoints in the group. You can optionally create application-specific policies and attach them to specific endpoints.
- **Trust providers** – A service that manages user identities or device security state. Verified Access works with both AWS and third-party trust providers. You must attach at least one trust provider to each Verified Access instance. You can attach a single identity trust provider and multiple device trust providers to each Verified Access instance.
- **Trust data** – The security-related data for users or devices that your trust provider sends to Verified Access. Also referred to as *user claims* or *trust context*. For example, the email address of a user or the operating system version of a device. Verified Access evaluates this data against your access policies when it receives each request to access an application.

Tutorial: Get started with Verified Access

Use this tutorial to get started with AWS Verified Access. You'll learn how to create and configure Verified Access resources.

As a part of this tutorial, you'll add an application to Verified Access. At the end of the tutorial, specific users can access that application over the internet, without using VPN. Instead, you'll use AWS IAM Identity Center as an identity trust provider. Note that this tutorial doesn't also use a device trust provider.

Tasks

- [Verified Access tutorial prerequisites](#)
- [Step 1: Create a Verified Access trust provider](#)
- [Step 2: Create a Verified Access instance](#)
- [Step 3: Create a Verified Access group](#)
- [Step 4: Create a Verified Access endpoint](#)
- [Step 5: Configure DNS for the Verified Access endpoint](#)
- [Step 6: Test connectivity to the application](#)
- [Step 7: Add a Verified Access group-level access policy](#)
- [Clean up your Verified Access resources](#)

Verified Access tutorial prerequisites

The following are the prerequisites for completing this tutorial:

- AWS IAM Identity Center enabled in the AWS Region that you're working in. You can then use IAM Identity Center as a trust provider with Verified Access. For more information, see [Enable AWS IAM Identity Center](#) in the *AWS IAM Identity Center User Guide*.
- A security group to control access to the application. Allow all inbound traffic from the VPC CIDR and all outbound traffic.
- An application running behind an internal load balancer from Elastic Load Balancing. Associate your security group with the load balancer.
- A self-signed or public TLS certificate in AWS Certificate Manager. Use an RSA certificate with a key length of 1,024 or 2,048.

- A public hosted domain and the permissions required to update DNS records for the domain.
- An IAM policy with the permissions required to create an AWS Verified Access instance. For more information, see [Policy for creating Verified Access instances](#).

Step 1: Create a Verified Access trust provider

Use the following procedure to set up AWS IAM Identity Center as your trust provider.

To create an IAM Identity Center trust provider

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access trust providers**.
3. Choose **Create Verified Access trust provider**.
4. (Optional) For **Name tag** and **Description**, enter a name and description for the Verified Access trust provider.
5. Enter a custom identifier to use later when working with policy rules for **Policy reference name**. For example, you can enter **idc**.
6. For **Trust provider type**, choose **User trust provider**.
7. For **User trust provider type**, choose **IAM Identity Center**.
8. Choose **Create Verified Access trust provider**.

Step 2: Create a Verified Access instance

Use the following procedure to create a Verified Access instance.

To create a Verified Access instance

1. In the navigation pane, choose **Verified Access instances**.
2. Choose **Create Verified Access instance**.
3. (Optional) For **Name** and **Description**, enter a name and description for the Verified Access instance.
4. For **Verified Access trust provider**, choose your trust provider.
5. Choose **Create Verified Access instance**.

Step 3: Create a Verified Access group

Use the following procedure to create a Verified Access group.

To create a Verified Access group

1. In the navigation pane, choose **Verified Access groups**.
2. Choose **Create Verified Access group**.
3. (Optional) For **Name tag** and **Description**, enter a name and description for the group.
4. For **Verified Access instance**, choose your Verified Access instance.
5. Keep **Policy definition** blank. You will add a group-level policy in a later step.
6. Choose **Create Verified Access group**.

Step 4: Create a Verified Access endpoint

Use the following procedure to create a Verified Access endpoint. This step assumes that you have an application running behind an internal load balancer from Elastic Load Balancing and a public domain certificate in AWS Certificate Manager.

To create a Verified Access endpoint

1. In the navigation pane, choose **Verified Access endpoints**.
2. Choose **Create Verified Access endpoint**.
3. (Optional) For **Name tag** and **Description**, enter a name and description for the endpoint.
4. For **Verified Access group**, choose your Verified Access group.
5. For **Endpoint details**, do the following:
 - a. For **Protocol**, select **HTTPS** or **HTTP**, depending on the configuration of your load balancer.
 - b. For **Attachment type**, choose **VPC**.
 - c. For **Endpoint type**, choose **Load balancer**.
 - d. For **Port**, enter the port number used by your load balancer listener. For example, 443 for HTTPS or 80 for HTTP.
 - e. For **Load balancer ARN**, choose your load balancer.

- f. For **Subnets**, select the subnets associated with your load balancer.
 - g. For **Security groups**, select your security group. Using the same security group for your load balancer and endpoint allows traffic between them. If you prefer not to use the same security group, be sure to reference the endpoint security group from your load balancer so that it accepts traffic from the endpoint.
 - h. For **Endpoint domain prefix**, enter a custom identifier. For example, **my-ava-app**. This prefix is prepended to the DNS name that Verified Access generates.
6. For **Application details**, do the following:
 - a. For **Application domain**, enter the DNS name for your application. This domain must match the one in your domain certificate.
 - b. For **Domain certificate ARN**, select the Amazon Resource Name (ARN) of your domain certificate in AWS Certificate Manager.
7. Keep **Policy details** blank. You will add a group-level access policy in a later step.
8. Choose **Create Verified Access endpoint**.

Step 5: Configure DNS for the Verified Access endpoint

For this step, you map your application's domain name (for example, `www.myapp.example.com`) to the domain name of your Verified Access endpoint. To complete the DNS mapping, create a Canonical Name Record (CNAME) with your DNS provider. After you create the CNAME record, all requests from users to your application will be sent to Verified Access.

To get the domain name of your endpoint

1. In the navigation pane, choose **Verified Access endpoints**.
2. Select your endpoint.
3. Choose the **Details** tab.
4. Copy the domain from **Endpoint domain**. The following is an example endpoint domain name: `my-ava-app.edge-1a2b3c4d5e6f7g.vai-1a2b3c4d5e6f7g.prod.verified-access.us-west-2.amazonaws.com`.

Follow the directions provided by your DNS provider to create a CNAME record. Use the domain name of your application as the record name and the domain name of the Verified Access endpoint as the record value.

Step 6: Test connectivity to the application

You can now test connectivity to your application. Enter your application's domain name into your web browser. The default behavior of Verified Access is to deny all requests. Because we did not add a Verified Access policy to the group or the endpoint, all requests are denied.

Step 7: Add a Verified Access group-level access policy

Use the following procedure to modify the Verified Access group and configure an access policy that allows connectivity to your application. The details of the policy will depend on the users and groups that are configured in IAM Identity Center. For information, see [Verified Access policies](#).

To modify a Verified Access group

1. In the navigation pane, choose **Verified Access groups**.
2. Select your group.
3. Choose **Actions, Modify Verified Access group policy**.
4. Turn on **Enable policy**.
5. Enter a policy that allows users from your IAM Identity Center to access your application. For examples, see [the section called "Example policies"](#).
6. Choose **Modify Verified Access group policy**.
7. Now that your group policy is in place, repeat the test from the previous step to verify that the request is allowed. If the request is allowed, you are prompted to sign in through the IAM Identity Center sign-in page. After you provide the user name and password, you can access your application.

Clean up your Verified Access resources

After you are finished with this tutorial, use the following procedure to delete your Verified Access resources.

To delete your Verified Access resources

1. In the navigation pane, choose **Verified Access endpoints**. Select the endpoint and choose **Actions, Delete Verified Access endpoint**.

2. In the navigation pane, choose **Verified Access groups**. Select the group and choose **Actions, Delete Verified Access group**. You might need to wait until the endpoint deletion process is complete.
3. In the navigation pane, choose **Verified Access instances**. Select your instance and choose **Actions, Detach Verified Access trust provider**. Select the trust provider and choose **Detach Verified Access trust provider**.
4. In the navigation pane, choose **Verified Access trust providers**. Select your trust provider and choose **Actions, Delete Verified Access trust provider**.
5. In the navigation pane, choose **Verified Access instances**. Select your instance and choose **Actions, Delete Verified Access instance**.

Verified Access instances

An AWS Verified Access instance is an AWS resource that helps you organize your trust providers and Verified Access groups. An instance evaluates application requests and grants access only when your security requirements are met.

Tasks

- [Create and manage a Verified Access instance](#)
- [Delete a Verified Access instance](#)
- [Integrate Verified Access with AWS WAF](#)
- [FIPS compliance for Verified Access](#)

Create and manage a Verified Access instance

You use a Verified Access instance to organize your trust providers and Verified Access groups. Use the following procedures to create a Verified Access instance, and then attach a trust provider to Verified Access or detach a trust provider from Verified Access.

Tasks

- [Create a Verified Access instance](#)
- [Attach a trust provider to a Verified Access instance](#)
- [Detach a trust provider from a Verified Access instance](#)
- [Add a custom subdomain](#)

Create a Verified Access instance

Use the following procedure to create a Verified Access instance.

To create a Verified Access instance using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access instances**, and then **Create Verified Access instance**.
3. (Optional) For **Name** and **Description**, enter a name and description for the Verified Access instance.

4. (Network CIDR endpoints) For **Custom subdomain for network CIDR endpoint**, enter a custom subdomain.
5. (Optional) Choose **Enable** for **Federal Information Process Standards (FIPS)** if you require Verified Access to be FIPS compliant.
6. (Optional) For **Verified Access trust provider**, choose a trust provider to attach to the Verified Access instance.
7. (Optional) To add a tag, choose **Add new tag** and enter the tag key and the tag value.
8. Choose **Create Verified Access instance**.

To create a Verified Access instance using the AWS CLI

Use the [create-verified-access-instance](#) command.

Attach a trust provider to a Verified Access instance

Use the following procedure to attach a trust provider to a Verified Access instance.

To attach a trust provider to a Verified Access instance using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access instances**.
3. Select the instance.
4. Choose **Actions, Attach Verified Access trust provider**.
5. For **Verified Access trust provider**, choose a trust provider.
6. Choose **Attach Verified Access trust provider**.

To attach a trust provider to a Verified Access instance using the AWS CLI

Use the [attach-verified-access-trust-provider](#) command.

Detach a trust provider from a Verified Access instance

Use the following procedure to detach a trust provider from a Verified Access instance.

To detach a trust provider from a Verified Access instance using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.

2. In the navigation pane, choose **Verified Access instances**.
3. Select the instance.
4. Choose **Actions, Detach Verified Access trust provider**.
5. For **Verified Access trust provider**, choose the trust provider.
6. Choose **Detach Verified Access trust provider**.

To detach a trust provider from a Verified Access instance using the AWS CLI

Use the [detach-verified-access-trust-provider](#) command.

Add a custom subdomain

Use the following procedure to add or update a custom subdomain. This subdomain is used only when you create a [network CIDR endpoint](#).

To add a custom subdomain using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access instances**.
3. Select the instance.
4. Choose **Actions, Modify Verified Access instance**.
5. For **Custom subdomain for network CIDR endpoint**, enter a custom subdomain.
6. Choose **Modify Verified Access instance**.
7. Update the nameservers for your subdomain, entering the nameservers provided by Verified Access. This list is available under **Nameservers** on the **Details** tab for the instance.

To add a custom subdomain using the AWS CLI

Use the [modify-verified-access-instance](#) command.

Delete a Verified Access instance

When you are finished with a Verified Access instance, you can delete it. Before you can delete an instance, you must remove any associated trust providers or Verified Access groups.

To delete a Verified Access instance using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access instances**.
3. Select the Verified Access instance.
4. Choose **Actions, Delete Verified Access instance**.
5. When prompted for confirmation, enter **delete**, and then choose **Delete**.

To delete a Verified Access instance using the AWS CLI

Use the [delete-verified-access-instance](#) command.

Integrate Verified Access with AWS WAF

In addition to the authentication and authorization rules enforced by Verified Access, you might also want to apply perimeter protection. This can help you protect your applications from additional threats. You can accomplish this by integrating AWS WAF into your Verified Access deployment. AWS WAF is a web application firewall that lets you monitor the HTTP requests that are forwarded to your protected web application resources. For more information, see the [AWS WAF Developer Guide](#).

You can integrate AWS WAF with Verified Access by associating an AWS WAF web access control list (ACL) with a Verified Access instance. A web ACL is a AWS WAF resource that gives you fine-grained control over all of the HTTP web requests that your protected resource responds to. While the AWS WAF association or disassociation request is being processed, the status of any Verified Access endpoints attached to the instance are shown as `updating`. After the request is complete, the status returns to `active`. You can view the status in the AWS Management Console or by describing the endpoint with the AWS CLI.

The user-identity trust provider determines when AWS WAF inspects the traffic. If you use IAM Identity Center, AWS WAF inspects the traffic before user authentication. If you use OpenID Connect (OIDC), AWS WAF inspects the traffic after user authentication.

Contents

- [Required IAM permissions](#)
- [Associate an AWS WAF web ACL](#)

- [Check the status of the association](#)
- [Disassociate an AWS WAF web ACL](#)

Required IAM permissions

Integrating AWS WAF with Verified Access includes permission-only actions that don't directly correspond to an API operation. These actions are indicated in the AWS Identity and Access Management *Service Authorization Reference* with [permission only]. See [Actions, resources, and condition keys for Amazon EC2](#) in the *Service Authorization Reference*.

To work with a web ACL, your AWS Identity and Access Management principal must have the following permissions.

- `ec2:AssociateVerifiedAccessInstanceWebAcl`
- `ec2:DisassociateVerifiedAccessInstanceWebAcl`
- `ec2:DescribeVerifiedAccessInstanceWebAclAssociations`
- `ec2:GetVerifiedAccessInstanceWebAcl`

Associate an AWS WAF web ACL

The following steps demonstrate how to associate an AWS WAF web access control list (ACL) with a Verified Access instance using the Verified Access console.

Prerequisite

Before you begin, create a AWS WAF web ACL. For more information, see [Create a web ACL](#) in the *AWS WAF Developer Guide*.

To associate an AWS WAF web ACL to a Verified Access instance

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access instances**.
3. Select the Verified Access instance.
4. Select the **Integrations** tab.
5. Choose **Actions**, then **Associate Web ACL**.
6. For **Web ACL**, choose an existing web ACL, then choose **Associate Web ACL**.

Alternatively, you can use the AWS WAF console. If you use the AWS WAF console or API, you need the Amazon Resource Name (ARN) of your Verified Access instance. An AVA ARN has the following format: `arn:${Partition}:ec2:${Region}:${Account}:verified-access-instance/${VerifiedAccessInstanceId}`. For more information, see [Associate a web ACL with an AWS resource](#) in the *AWS WAF Developer Guide*.

Check the status of the association

You can verify whether an AWS WAF web access control list (ACL) is associated with a Verified Access instance or not by using the Verified Access console.

To view the status of AWS WAF integration with a Verified Access instance

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access instances**.
3. Select the Verified Access instance.
4. Select the **Integrations** tab.
5. Check the details listed under **WAF integration status**. The status will be shown as **Associated** or **Not associated**, along with the web ACL identifier, if in the **Associated** state.

Disassociate an AWS WAF web ACL

The following steps demonstrate how to disassociate an AWS WAF web access control list (ACL) from a Verified Access instance using the Verified Access console.

To disassociate an AWS WAF web ACL from a Verified Access instance

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access instances**.
3. Select the Verified Access instance.
4. Select the **Integrations** tab.
5. Choose **Actions**, then **Disassociate Web ACL**.
6. Confirm by choosing **Disassociate Web ACL**.

Alternatively, you can use the AWS WAF console. For more information, see [Disassociate a web ACL from an AWS resource](#) in the *AWS WAF Developer Guide*.

FIPS compliance for Verified Access

Federal Information Processing Standard (FIPS) is a US and Canadian government standard that specifies security requirements for cryptographic modules that protect sensitive information. AWS Verified Access provides the option to configure your environment to adhere to FIPS Publication 140-2. FIPS compliance for Verified Access is available in the following AWS Regions:

- US East (Ohio)
- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- Canada (Central)
- AWS GovCloud (US) West
- AWS GovCloud (US) East

This page shows you how to configure a new, or an existing Verified Access environment, to be FIPS compliant.

Contents

- [Configure an existing Verified Access environment for FIPS compliance](#)
- [Configure a new Verified Access environment for FIPS compliance](#)

Configure an existing Verified Access environment for FIPS compliance

If you have an existing Verified Access environment and you want to configure it to be FIPS compliant, some of the resources will need to be deleted and re-created in order to turn on FIPS compliance.

To re-configure an existing AWS Verified Access environment to be FIPS compliant, follow the steps below.

1. Delete your original Verified Access endpoint(s), group(s), and instance. Your configured trust providers can be re-used.
2. Create a Verified Access instance, making sure to enable **Federal Information Process Standards (FIPS)** during creation. Also during creation, attach the **Verified Access trust provider** you want to use, by selecting it from the drop down list.

3. Create a Verified Access [group](#). During creation of the group, you associate it with the Verified Access instance just created.
4. Create one or more [Verified Access endpoints](#). During the creation of your endpoint(s), you associate them with the group created in the previous step.

Configure a new Verified Access environment for FIPS compliance

To configure a new AWS Verified Access environment that is FIPS compliant, follow the steps below.

1. Configure a [trust provider](#). You will need to create a [user identity](#) trust provider and (optionally) a [device-based](#) trust provider, depending on your needs.
2. Create a Verified Access [instance](#), making sure to enable **Federal Information Process Standards (FIPS)** during the process. Also during creation, attach the **Verified Access trust provider** you created in the previous step, by selecting it from the drop down list.
3. Create a Verified Access [group](#). During creation of the group, you associate it with the Verified Access instance just created.
4. Create one or more [Verified Access endpoints](#). During the creation of your endpoint(s), you associate them with the group created in the previous step.

Trust providers for Verified Access

A trust provider is a service that sends information about users and devices to AWS Verified Access. This information is called trust context. It can include attributes based on user identity, such as an email address or membership in the "sales" organization, or device information such as installed security patches or anti-virus software version.

Verified Access supports the following categories of trust providers:

- **User identity** – An identity provider (IdP) service that stores and manages digital identities for users.
- **Device management** – A device management system for devices such as laptops, tablets, and smartphones.

Contents

- [User-identity trust providers for Verified Access](#)
- [Device-based trust providers for Verified Access](#)

User-identity trust providers for Verified Access

You can choose to use either AWS IAM Identity Center or an OpenID Connect-compatible user-identity trust provider.

Contents

- [Using IAM Identity Center as a trust provider](#)
- [Use an OpenID Connect trust provider](#)

Using IAM Identity Center as a trust provider

You can use AWS IAM Identity Center as your *user-identity* trust provider with AWS Verified Access.

Prerequisites and considerations

- Your IAM Identity Center instance must be an AWS Organizations instance. A standalone AWS account IAM Identity Center instance will not work.

- Your IAM Identity Center instance must be enabled in the same AWS Region that you want to create the Verified Access trust provider in.
- Verified Access can provide access to users in IAM Identity Center who are assigned to up to 1,000 groups.

See [Manage organization and account instances of IAM Identity Center](#) in the *AWS IAM Identity Center User Guide* for details on the different instance types.

Create an IAM Identity Center trust provider

After IAM Identity Center is enabled on your AWS account, you can use the following procedure to set up IAM Identity Center as your trust provider for Verified Access.

To create an IAM Identity Center trust provider (AWS console)

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access trust providers**, and then **Create Verified Access trust provider**.
3. (Optional) For **Name tag** and **Description**, enter a name and description for the trust provider.
4. For **Policy reference name**, enter an identifier to use later when working with policy rules.
5. Under **Trust provider type**, select **User trust provider**.
6. Under **User trust provider type**, select **IAM Identity Center**.
7. (Optional) To add a tag, choose **Add new tag** and enter the tag key and the tag value.
8. Choose **Create Verified Access trust provider**.

To create an IAM Identity Center trust provider (AWS CLI)

- [create-verified-access-trust-provider](#) (AWS CLI)

Delete an IAM Identity Center trust provider

Before you can delete a trust provider, you must remove all endpoint and group configuration from the instance to which the trust provider is attached.

To delete an IAM Identity Center trust provider (AWS console)

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.

2. In the navigation pane, choose **Verified Access trust providers**, and then select the trust provider you want to delete under **Verified Access trust providers**.
3. Choose **Actions**, then **Delete Verified Access trust provider**.
4. Confirm the deletion by entering `delete` into the text box.
5. Choose **Delete**.

To delete an IAM Identity Center trust provider (AWS CLI)

- [delete-verified-access-trust-provider](#) (AWS CLI)

Use an OpenID Connect trust provider

AWS Verified Access supports identity providers that use standard OpenID Connect (OIDC) methods. You can use OIDC compatible providers as *user-identity* trust providers with Verified Access. However, due to the wide array of potential OIDC providers, AWS is not able to test each OIDC integration with Verified Access.

Verified Access obtains the trust data that it evaluates from the OIDC provider's `UserInfo` Endpoint. The `Scope` parameter is used to determine which sets of trust data will be retrieved. After the trust data is received, the Verified Access policy is evaluated against it.

The ID token claims from the OIDC trust provider are included in the `addition_user_context` key, for trust providers created after February 24, 2025.

With trust providers created on or before February 24, 2025, Verified Access does not use trust data from the ID token sent by the OIDC provider. Only trust data from the `UserInfo` Endpoint is evaluated against the policy.

With trust providers created on or before February 24, 2025, the default session duration is one day. With trust providers created before February 24, 2025, the default session duration is seven days.

If a refresh token is specified, Verified Access uses the expiration of the refresh token as the session duration. If there is no refresh token, the default session duration is used.

Contents

- [Prerequisites for creating an OIDC trust provider](#)

- [Create an OIDC trust provider](#)
- [Modify an OIDC trust provider](#)
- [Delete an OIDC trust provider](#)

Prerequisites for creating an OIDC trust provider

You will need to gather the following information from your trust provider service directly:

- Issuer
- Authorization endpoint
- Token endpoint
- UserInfo endpoint
- Client ID
- Client secret
- Scope

Create an OIDC trust provider

Use the following procedure to create an OIDC as your trust provider.

To create an OIDC trust provider (AWS console)

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access trust providers**, and then **Create Verified Access trust provider**.
3. (Optional) For **Name tag** and **Description**, enter a name and description for the trust provider.
4. For **Policy reference name**, enter an identifier to use later when working with policy rules.
5. Under **Trust provider type**, select **User trust provider**.
6. Under **User trust provider type**, select **OIDC (OpenID Connect)**.
7. For **OIDC (OpenID Connect)**, choose the trust provider.
8. For **Issuer**, enter the identifier of the OIDC issuer.
9. For **Authorization endpoint**, enter the full URL of the authorization endpoint.
10. For **Token endpoint**, enter the full URL of the token endpoint.
11. For **User endpoint**, enter the full URL of the user endpoint.

12. (Native Application OIDC) For **Public signing key URL**, enter the full URL of the public signing key endpoint.
13. Enter the OAuth 2.0 client identifier for **Client ID**.
14. Enter the OAuth 2.0 client secret for **Client secret**.
15. Enter a space-delimited list of scopes defined with your identity provider. At minimum, the openid scope is required for **Scope**.
16. (Optional) To add a tag, choose **Add new tag** and enter the tag key and the tag value.
17. Choose **Create Verified Access trust provider**.
18. You must add a redirect URI to the allow list for your OIDC provider.
 - HTTP applications – Use the following URI: **https://application_domain/oauth2/idpresponse**. In the console, you can find the application domain on the **Details** tab for the Verified Access endpoint. Using the AWS CLI or an AWS SDK, the application domain is included in the output when you describe the Verified Access endpoint.
 - TCP applications – Use the following URI: **http://localhost:8000**.

To create an OIDC trust provider (AWS CLI)

- [create-verified-access-trust-provider](#) (AWS CLI)

Modify an OIDC trust provider

After you create a trust provider, you can update its configuration.

To modify an OIDC trust provider (AWS console)

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access trust providers**, and then select the trust provider you want to modify under **Verified Access trust providers**.
3. Choose **Actions**, then **Modify Verified Access trust provider**.
4. Modify the options you want to change.
5. Choose **Modify Verified Access trust provider**.

To modify an OIDC trust provider (AWS CLI)

- [modify-verified-access-trust-provider](#) (AWS CLI)

Delete an OIDC trust provider

Before you can delete a user trust provider, you first need to remove all endpoint and group configuration from the instance the trust provider is attached to.

To delete an OIDC trust provider (AWS console)

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access trust providers**, and then select the trust provider you want to delete under **Verified Access trust providers**.
3. Choose **Actions**, then **Delete Verified Access trust provider**.
4. Confirm the deletion by entering delete into the text box.
5. Choose **Delete**.

To delete an OIDC trust provider (AWS CLI)

- [delete-verified-access-trust-provider](#) (AWS CLI)

Device-based trust providers for Verified Access

You can use device trust providers with AWS Verified Access. You can use one or multiple device trust providers with your Verified Access instance.

Contents

- [Supported device trust providers](#)
- [Create a device-based trust provider](#)
- [Modify a device-based trust provider](#)
- [Delete a device-based trust provider](#)

Supported device trust providers

The following device trust providers can be integrated with Verified Access:

- CrowdStrike – [Securing private applications with CrowdStrike and AWS Verified Access](#)
- Jamf – [Integrating Verified Access with Jamf Device Identity](#)
- JumpCloud – [Integrating JumpCloud and AWS Verified Access](#)

Create a device-based trust provider

Follow these steps to create and configure a device trust provider to use with Verified Access.

To create a Verified Access device trust provider (AWS console)

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access trust providers**, and then **Create Verified Access trust provider**.
3. (Optional) For **Name tag** and **Description**, enter a name and description for the trust provider.
4. Enter an identifier to use later when working with policy rules for **Policy reference name**.
5. For **Trust provider type**, select **Device identity**.
6. For **Device identity type**, choose **Jamf**, **CrowdStrike**, or **JumpCloud**.
7. For **Tenant ID**, enter the identifier of the tenant application.
8. (Optional) For **Public signing key URL**, enter the unique key URL shared by your device trust provider. (This parameter is not required for Jamf, CrowdStrike or Jumpcloud.)
9. Choose **Create Verified Access trust provider**.

Note

You will need to add a redirect URI to your OIDC provider's allowlist. You will want to use the `DeviceValidationDomain` of the Verified Access endpoint for this purpose. This can be found in the AWS Management Console, under the **Details** tab for your Verified Access endpoint or by using the AWS CLI to describe the endpoint. Add the following to your OIDC provider's allowlist: `https://DeviceValidationDomain/oauth2/idpresponse`

To create a Verified Access device trust provider (AWS CLI)

- [create-verified-access-trust-provider](#) (AWS CLI)

Modify a device-based trust provider

After you create a trust provider, you can update its configuration.

To modify a Verified Access device trust provider (AWS console)

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access trust providers**.
3. Select the trust provider.
4. Choose **Actions**, then select **Modify Verified Access trust provider**.
5. Modify the description as needed.
6. (Optional) For **Public signing key URL**, modify the unique key URL shared by your device trust provider. (This parameter is not required if your device trust provider is Jamf, CrowdStrike or Jumpcloud.)
7. Choose **Modify Verified Access trust provider**.

To modify a Verified Access device trust provider (AWS CLI)

- [modify-verified-access-trust-provider](#) (AWS CLI)

Delete a device-based trust provider

When you are finished with a trust provider, you can delete it.

To delete a Verified Access device trust provider (AWS console)

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access trust providers**.
3. Select the trust provider you want to delete under **Verified Access trust providers**.
4. Choose **Actions**, then select **Delete Verified Access trust provider**.
5. When prompted for confirmation, enter **delete**, and then choose **Delete**.

To delete a Verified Access device trust provider (AWS CLI)

- [delete-verified-access-trust-provider](#) (AWS CLI)

Verified Access groups

A Verified Access group consists of Verified Access endpoints and a Verified Access policy that applies to all endpoints in the group. By grouping together endpoints that have common security requirements, you can define a single group policy that meets the minimum security requirements of multiple endpoints. Therefore, you don't need create and maintain a policy for each endpoint.

For example, you can group all sales applications together and set a group-wide access policy. You can then use this policy to define a common set of minimum security requirements for all sales applications. This approach helps to simplify policy administration.

When you create a group, you are required to associate the group with a Verified Access instance. During the process of creating an endpoint, you will associate the endpoint with a group.

Another feature of Verified Access groups is the ability to share them with other AWS accounts using AWS RAM. This allows you to create and manage groups centrally in one account, then share them with multiple accounts.

Tasks

- [Create and manage a Verified Access group](#)
- [Modify a Verified Access group policy](#)
- [Share a Verified Access group with another AWS account](#)
- [Delete a Verified Access group](#)

Create and manage a Verified Access group

You use Verified Access groups to organize endpoints by their security requirements. When you create a Verified Access endpoint, you associate the endpoint with a group.

Tasks

- [Create a Verified Access group](#)
- [Modify a Verified Access group](#)

Create a Verified Access group

Use the following procedures to create a Verified Access group. Before you create a Verified Access group, you must create a Verified Access instance. For more information, see [the section called "Create a Verified Access instance"](#).

To create a Verified Access group using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access groups**, and then **Create Verified Access group**.
3. (Optional) For **Name tag** and **Description**, enter a name and description for the group.
4. For **Verified Access instance**, select a Verified Access instance to associate with the group.
5. (Optional) For **Policy definition**, enter a Verified Access policy to apply to the group.
6. (Optional) To add a tag, choose **Add new tag** and enter the tag key and the tag value.
7. Choose **Create Verified Access group**.

To create a Verified Access group using the AWS CLI

Use the [create-verified-access-group](#) command.

Modify a Verified Access group

Use the following procedure to modify a Verified Access group.

To modify a Verified Access group using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access groups**, and then **Create Verified Access group**.
3. Select the group and then choose **Actions, Modify Verified Access group**.
4. (Optional) Update the description.
5. Choose **Create Verified Access group**.
6. Choose the Verified Access instance to associate with the group.

To modify a Verified Access group using the AWS CLI

Use the [modify-verified-access-group](#) command.

Modify a Verified Access group policy

AWS Verified Access allows access to your applications based on the access policies that you create. The Verified Access policy that you attach to a group is inherited by all endpoints in the group. You can optionally attach application-specific policies to specific endpoints.

Use the following procedure to modify the policy for a Verified Access group. After you make the changes, it takes several minutes before they take effect.

To modify a Verified Access group policy using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access groups**.
3. Select the group.
4. Choose **Actions, Modify Verified Access group policy**.
5. (Optional) Turn on or off **Enable policy** as needed.
6. (Optional) For **Policy**, enter the Verified Access policy to apply to the group.
7. Choose **Modify Verified Access group policy**.

To modify a Verified Access group policy using the AWS CLI

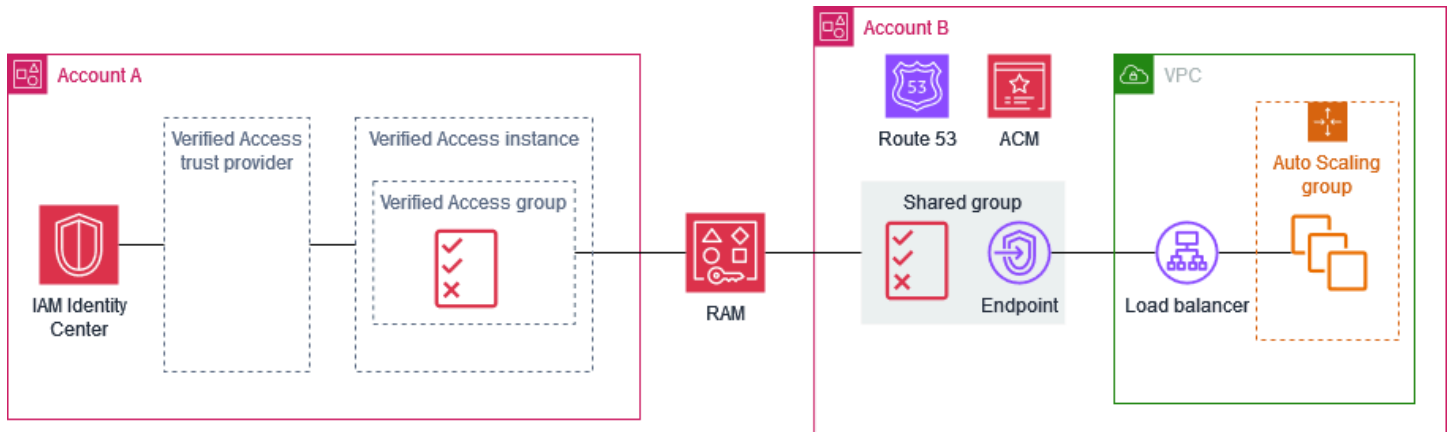
Use the [modify-verified-access-group-policy](#) command.

Share a Verified Access group with another AWS account

When you share a Verified Access group that you own with other AWS accounts, you enable those accounts to create Verified Access endpoints in your group. The account that created the Verified Access group is referred to as the *owner* account. The account that uses a shared group is referred to as the *consumer* account.

The following diagram illustrates the benefit of sharing a Verified Access group. The central security team owns Account A. They manage users and groups in AWS IAM Identity Center, and manage the Verified Access resources required to provide access to internal applications, such as

Verified Access trust providers, Verified Access instances, Verified Access groups, and Verified Access policies. The application team owns Account B. They manage the resources required to run their internal application, such as the load balancer, Auto Scaling group, DNS configuration in Amazon Route 53, and TLS certificates from AWS Certificate Manager (ACM). After the central security team shares a Verified Access group with Account B, the application team can create Verified Access endpoints using the shared group. Access to the application is allowed or denied based on the policies that the central security team created for the Verified Access group.



Considerations

The following considerations apply to shared Verified Access groups.

Owners

- To share a Verified Access group, users must have the following permissions: `ec2:PutResourcePolicy` and `ec2>DeleteResourcePolicy`.
- To share a Verified Access group, you must own it. You can't share a Verified Access group that was shared with you.
- If you enable sharing with the accounts in your organization, you can share resources, such as Verified Access groups, without using invitations. Otherwise, the consumer receives an invitation and must accept it to access the shared group. To enable sharing, from the management account for your organization, open the [Settings](#) page in the AWS RAM console and choose **Enable sharing with AWS Organizations**.
- You can't delete a group if there are associated Verified Access endpoints. You can view the endpoints created by consumer accounts on the **Verified Access endpoints** page in your account. The account ID of the owner of an endpoint is reflected in the Amazon Resource Name (ARN) of the certificate for the endpoint.

Consumers

- To view the Verified Access groups that are shared with you, open the **Verified Access groups** page in the console, or call [describe-verified-access-groups](#). The account ID of the owner is reflected in the **Owner** field and the Amazon Resource Name (ARN) of the group.
- When you create a Verified Access endpoint, you can specify any Verified Access groups that were shared with you.
- You can't view endpoints that are associated with the shared group but not owned by you.
- If the owner of the Verified Access group deletes the resource share, you can't create a new Verified Access endpoint in the group. Any Verified Access endpoints that you created prior to the deletion of the resource share are unaffected by the deletion of the resource share. However, the owner of the shared group can delete your endpoints.

Resource shares

To share a Verified Access group, you must add it to a resource share. A resource share specifies the resources to share and the consumers that can use the shared resources.

To share a Verified Access group using the console

1. Open the AWS RAM console at <https://console.aws.amazon.com/ram/home>.
2. If you don't have a resource share for your organization, create one. For the principal, you can choose your entire organization, an organizational unit, or specific AWS accounts.
3. Select your resource share and choose **Modify**.
4. For **Resources**, choose **Verified Access Groups** as the resource type, and then select the resource group to share.
5. Choose **Skip to: Review and update**.
6. Choose **Update resource share**.

For more information, see [Create a resource share](#) in the *AWS RAM User Guide*.

Delete a Verified Access group

When you are finished with a Verified Access group, you can delete it. You can't delete a group if there are associated Verified Access endpoints.

To delete a Verified Access group using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access groups**.
3. Select the group.
4. Choose **Actions, Delete Verified Access group**.
5. When prompted for confirmation, enter **delete**, and then choose **Delete**.

To delete a Verified Access group using the AWS CLI

Use the [delete-verified-access-group](#) command.

Verified Access endpoints

A Verified Access endpoint represents an application. Each endpoint is associated with a Verified Access group and inherits the access policy for the group. You can optionally attach an application-specific endpoint policy to each endpoint.

Contents

- [Verified Access endpoint types](#)
- [How Verified Access works with shared VPCs and subnets](#)
- [Create a load balancer endpoint for Verified Access](#)
- [Create a network interface endpoint for Verified Access](#)
- [Create a network CIDR endpoint for Verified Access](#)
- [Create an Amazon Relational Database Service endpoint for Verified Access](#)
- [Allow traffic that originates from your Verified Access endpoint](#)
- [Modify a Verified Access endpoint](#)
- [Modify a Verified Access endpoint policy](#)
- [Delete a Verified Access endpoint](#)

Verified Access endpoint types

The following are the possible Verified Access endpoint types:

- **Load balancer** – Application requests are sent to a load balancer to distribute to your application. For more information, see [Create a load balancer endpoint](#).
- **Network interface** – Application requests are sent to a network interface using the specified protocol and port. For more information, see [Create a network interface endpoint](#).
- **Network CIDR** – Application requests are sent to the specified CIDR block. For more information, see [Create a network CIDR endpoint](#).
- **Amazon Relational Database Service (RDS)** – Application requests are sent to an RDS instance, RDS cluster, or RDS DB proxy. For more information, see [Create an Amazon Relational Database Service endpoint](#).

How Verified Access works with shared VPCs and subnets

The following are the behaviors regarding shared VPC subnets:

- Verified Access endpoints are supported by VPC subnet sharing. A participant can create a Verified Access endpoint in a shared subnet.
- The participant who created the endpoint will be the endpoint owner, and the only party allowed to modify the endpoint. The VPC owner will not be allowed to modify the endpoint.
- Verified Access endpoints cannot be created in an AWS Local Zone and therefore sharing via Local Zones is not possible.

For more information see, [Share your VPC with other accounts](#) in the *Amazon VPC User Guide*.

Create a load balancer endpoint for Verified Access

Use the following procedure to create a load balancer endpoint for Verified Access. For more information about load balancers, see the [Elastic Load Balancing User Guide](#).

Requirements

- Only IPv4 traffic is supported.
- Long-lived HTTPS connections, such as WebSocket connections, are supported only through TCP.
- The load balancer must be either an Application Load Balancer or a Network Load Balancer, and it must be an internal load balancer.
- The load balancer and subnets must belong to the same virtual private cloud (VPC).
- HTTPS load balancers can use either self-signed or public TLS certificates. Use an RSA certificate with a key length of 1,024 or 2,048.
- Before you create a Verified Access endpoint, you must create a Verified Access group. For more information, see [the section called “Create a Verified Access group”](#).
- You must provide a domain name for your application. This is the public DNS name your users will use to access your application. You will also need to provide a public SSL certificate with a CN that matches this domain name. You can create or import the certificate using AWS Certificate Manager.

To create a load balancer endpoint using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access endpoints**.
3. Choose **Create Verified Access endpoint**.
4. (Optional) For **Name tag** and **Description**, enter a name and description for the endpoint.
5. For **Verified Access group**, choose a Verified Access group.
6. For **Endpoint details**, do the following:
 - a. For **Protocol**, choose a protocol.
 - b. For **Attachment type**, choose **VPC**.
 - c. For **Endpoint type**, choose **Load balancer**.
 - d. (HTTP/HTTPS) For **Port**, enter the port number. (TCP) For **Port ranges**, enter a port range and choose **Add port**.
 - e. For **Load balancer ARN**, choose a load balancer.
 - f. For **Subnet**, choose the subnets. You can specify only one subnet per Availability Zone.
 - g. For **Security groups**, choose the security groups for the endpoint. These security groups control the inbound and outbound traffic for the Verified Access endpoint.
 - h. For **Endpoint domain prefix**, enter a custom identifier to prepend to the DNS name that Verified Access generates for the endpoint.
7. (HTTP/HTTPS) For **Application details**, do the following:
 - a. For **Application domain**, enter a DNS name for your application.
 - b. Under **Domain certificate ARN**, choose a public TLS certificate.
8. (Optional) For **Policy definition**, enter a Verified Access policy for the endpoint.
9. (Optional) To add a tag, choose **Add new tag** and enter the tag key and the tag value.
10. Choose **Create Verified Access endpoint**.

To create a Verified Access endpoint using the AWS CLI

Use the [create-verified-access-endpoint](#) command.

Create a network interface endpoint for Verified Access

Use the following procedure to create a network interface endpoint.

Requirements

- Only IPv4 traffic is supported.
- The network interface must belong to the same virtual private cloud (VPC) as the security groups.
- We use the private IP on the network interface to forward the traffic.
- Before you create a Verified Access endpoint, you must create a Verified Access group. For more information, see [the section called "Create a Verified Access group"](#).
- You must provide a domain name for your application. This is the public DNS name your users will use to access your application. You will also need to provide a public SSL certificate with a CN that matches this domain name. You can create or import the certificate using AWS Certificate Manager.

To create a network interface endpoint using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access endpoints**.
3. Choose **Create Verified Access endpoint**.
4. (Optional) For **Name tag** and **Description**, enter a name and description for the endpoint.
5. For **Verified Access group**, choose a Verified Access group.
6. For **Endpoint details**, do the following:
 - a. For **Protocol**, choose a protocol.
 - b. For **Attachment type**, choose **VPC**.
 - c. For **Endpoint type**, choose **Network interface**.
 - d. (HTTP/HTTPS) For **Port**, enter the port number. (TCP) For **Port ranges**, enter a port range and choose **Add port**.
 - e. For **Network interface**, choose a network interface.
 - f. For **Security groups**, choose the security groups for the endpoint. These security groups control the inbound and outbound traffic for the Verified Access endpoint.

- g. For **Endpoint domain prefix**, enter a custom identifier to prepend to the DNS name that Verified Access generates for the endpoint.
7. (HTTP/HTTPS) For **Application details**, do the following:
 - a. For **Application domain**, enter a DNS name for your application.
 - b. Under **Domain certificate ARN**, choose a public TLS certificate.
8. (Optional) For **Policy definition**, enter a Verified Access policy for the endpoint.
9. (Optional) To add a tag, choose **Add new tag** and enter the tag key and the tag value.
10. Choose **Create Verified Access endpoint**.

To create a Verified Access endpoint using the AWS CLI

Use the [create-verified-access-endpoint](#) command.

Create a network CIDR endpoint for Verified Access

Use the following procedure to create a network CIDR endpoint. For example, you can use a network CIDR endpoint to enable access to EC2 instances in a specific subnet over port 22 (SSH).

Requirements

- Only the TCP protocol is supported.
- Verified Access provides a DNS record for each IP address in the CIDR range that is used by a resource. If you delete a resource, its IP address is no longer in use and Verified Access deletes the corresponding DNS record.
- If you specify a custom subdomain, Verified Access provides DNS records for each IP address in use in the subdomain and provides you with the IP addresses of its DNS servers. You can configure a forwarding rule for your subdomain to point to the Verified Access DNS servers. Any request made to a record in the domain is resolved by the Verified Access DNS servers to the IP address of the requested resource.
- Before you create a Verified Access endpoint, you must create a Verified Access group. For more information, see [the section called "Create a Verified Access group"](#).
- Create the endpoint and then connect to the application using the [Connectivity Client](#).

To create a network CIDR endpoint using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access endpoints**.
3. Choose **Create Verified Access endpoint**.
4. (Optional) For **Name tag** and **Description**, enter a name and description for the endpoint.
5. For **Verified Access group**, choose a Verified Access group for the endpoint.
6. For **Endpoint details**, do the following:
 - a. For **Protocol**, choose **TCP**.
 - b. For **Attachment type**, choose **VPC**.
 - c. For **Endpoint type**, choose **Network CIDR**.
 - d. For **Port ranges**, enter a port range and choose **Add port**.
 - e. For **Subnet**, choose the subnets.
 - f. For **Security groups**, choose the security groups for the endpoint. These security groups control the inbound and outbound traffic for the Verified Access endpoint.
 - g. (Optional) For **Endpoint domain prefix**, enter a custom identifier to prepend to the DNS name that Verified Access generates for the endpoint.
7. (Optional) For **Policy definition**, enter a Verified Access policy for the endpoint.
8. (Optional) To add a tag, choose **Add new tag** and enter the tag key and the tag value.
9. Choose **Create Verified Access endpoint**.

To create a Verified Access endpoint using the AWS CLI

Use the [create-verified-access-endpoint](#) command.

Create an Amazon Relational Database Service endpoint for Verified Access

Use the following procedure to create an Amazon Relational Database Service (RDS) endpoint.

Requirements

- Only the TCP protocol is supported.

- Create an RDS instance, RDS cluster, or RDS DB proxy.
- Before you create a Verified Access endpoint, you must create a Verified Access group. For more information, see [the section called “Create a Verified Access group”](#).
- Create the endpoint and then connect to the application using the [Connectivity Client](#).

To create an Amazon Relational Database Service endpoint using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access endpoints**.
3. Choose **Create Verified Access endpoint**.
4. (Optional) For **Name tag** and **Description**, enter a name and description for the endpoint.
5. For **Verified Access group**, choose a Verified Access group for the endpoint.
6. For **Endpoint details**, do the following:
 - a. For **Protocol**, choose **TCP**.
 - b. For **Attachment type**, choose **VPC**.
 - c. For **Endpoint type**, choose **Amazon Relational Database Service (RDS)**.
 - d. For **RDS target type**, do one of the following:
 - Choose **RDS instance**, and then choose an RDS instance from **RDS instance**.
 - Choose **RDS cluster**, and then choose an RDS cluster from **RDS cluster**.
 - Choose **RDS DB proxy**, and then choose an RDS DB proxy from **RDS DB proxy**.
 - e. For **RDS endpoint**, choose an RDS endpoint related to the RDS resource you chose in the previous step.
 - f. For **Port**, enter the port number.
 - g. For **Subnet**, choose the subnets. You can specify only one subnet per Availability Zone.
 - h. For **Security groups**, choose the security groups for the endpoint. These security groups control the inbound and outbound traffic for the Verified Access endpoint.
 - i. (Optional) For **Endpoint domain prefix**, enter a custom identifier to prepend to the DNS name that Verified Access generates for the endpoint.
7. (Optional) For **Policy definition**, enter a Verified Access policy for the endpoint.
8. (Optional) To add a tag, choose **Add new tag** and enter the tag key and the tag value.
9. Choose **Create Verified Access endpoint**.

To create a Verified Access endpoint using the AWS CLI

Use the [create-verified-access-endpoint](#) command.

Allow traffic that originates from your Verified Access endpoint

You can configure the security groups for your applications so that they allow traffic that originates from your Verified Access endpoint. You do this by adding an inbound rule that specifies the security group for the endpoint as the source. We recommend that you remove any additional inbound rules, so that your application receives traffic only from your Verified Access endpoint.

We recommend that you keep your existing outbound rules.

To update the security group rules for your application using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access endpoints**.
3. Choose the Verified Access endpoint, find **Security group IDs** on the **Details** tab, and copy the ID of the security group for your endpoint.
4. In the navigation pane, choose **Security groups**.
5. Select the check box for the security group associated with your target, and then choose **Actions, Edit inbound rules**.
6. To add a security group rule that allows traffic that originates from your Verified Access endpoint, do the following:
 - a. Choose **Add rule**.
 - b. For **Type**, choose **All traffic** or the specific traffic to allow.
 - c. For **Source**, choose **Custom** and paste the ID of the security group for your endpoint.
7. (Optional) To require that traffic originates only from your Verified Access endpoint, delete any other inbound security group rules.
8. Choose **Save rules**.

To update the security group rules for your application using the AWS CLI

Use the [describe-verified-access-endpoints](#) command to get the ID of the security group and then use the [authorize-security-group-ingress](#) command to add an inbound rule.

Modify a Verified Access endpoint

Use the following procedure to modify a Verified Access endpoint.

To modify a Verified Access endpoint using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access endpoints**.
3. Select the endpoint.
4. Choose **Actions, Modify Verified Access endpoint**.
5. Modify the endpoint details as needed.
6. Choose **Modify Verified Access endpoint**.

To modify a Verified Access endpoint using the AWS CLI

Use the [modify-verified-access-endpoint](#) command.

Modify a Verified Access endpoint policy

Use the following procedures to modify the policy for a Verified Access endpoint. After you make the changes, it takes several minutes before they take effect.

To modify a Verified Access endpoint policy using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access endpoints**.
3. Select the endpoint.
4. Choose **Actions, Modify Verified Access endpoint policy**.
5. (Optional) Turn on or off **Enable policy** as needed.
6. (Optional) For **Policy**, enter the Verified Access policy to apply to the endpoint.
7. Choose **Modify Verified Access endpoint policy**.

To modify a Verified Access endpoint policy using the AWS CLI

Use the [modify-verified-access-endpoint-policy](#) command.

Delete a Verified Access endpoint

When you are finished with a Verified Access endpoint, you can delete it.

To delete a Verified Access endpoint using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access endpoints**.
3. Select the endpoint.
4. Choose **Actions, Delete Verified Access endpoint**.
5. When prompted for confirmation, enter **delete** and then choose **Delete**.

To delete a Verified Access endpoint using the AWS CLI

Use the [delete-verified-access-endpoint](#) command.

Trust data sent to Verified Access from trust providers

Trust data is data sent to AWS Verified Access from a trust provider. Trust data is also referred to as "user claims" or "trust context." The data generally includes information about either a user or a device. Examples of trust data include user email, group membership, device operating system version, device security state, and so on. The information that's sent varies depending on the trust provider, so you should refer to your trust provider's documentation for a complete and updated list of trust data.

However, by using the Verified Access logging capabilities, you can also see what trust data is being sent from your trust provider. This can be useful when defining policies that allow or deny access to your applications. For information on including trust context in your logs, see [Enable or disable Verified Access trust context](#).

This section contains sample trust data and examples to help you get started with policy writing. The information provided here is intended for illustrative purposes only and not as an official reference.

Contents

- [Default context for Verified Access trust data](#)
- [AWS IAM Identity Center context for Verified Access trust data](#)
- [Third-party trust provider context for Verified Access trust data](#)
- [User claims passing and signature verification in Verified Access](#)

Default context for Verified Access trust data

AWS Verified Access includes some elements about the current request by default in all Cedar evaluations regardless of your configured trust providers. You can write a policy that evaluates against the data if you choose.

The following are examples of the data that is included in the evaluation.

Examples

- [HTTP request](#)
- [TCP flow](#)

HTTP request

When a policy is evaluated, Verified Access includes data about the current HTTP request in the Cedar context under the `context.http_request` key.

```
{
  "title": "HTTP Request data included by Verified Access",
  "type": "object",
  "properties": {
    "http_method": {
      "type": "string",
      "description": "The HTTP method",
      "example": "GET"
    },
    "hostname": {
      "type": "string",
      "description": "The host subcomponent of the authority component of the
URI",
      "example": "example.com"
    },
    "path": {
      "type": "string",
      "description": "The path component of the URI",
      "example": "app/images"
    },
    "query": {
      "type": "string",
      "description": "The query component of the URI",
      "example": "value1=1&value2=2"
    },
    "x_forwarded_for": {
      "type": "string",
      "description": "The value of the X-Forwarded-For request header",
      "example": "17.7.7.1"
    },
    "port": {
      "type": "integer",
      "description": "The endpoint port",
      "example": 443
    },
    "user_agent": {
      "type": "string",
      "description": "The value of the User-Agent request header",

```

```

        "example": "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0)
        Gecko/20100101 Firefox/47.0"
    },
    "client_ip": {
        "type": "string",
        "description": "The IP address connecting to the endpoint",
        "example": "15.248.6.6"
    }
}

```

Policy example

The following is an example Cedar policy that uses the HTTP request data.

```

forbid(principal, action, resource) when {
    context.http_request.http_method == "POST"
    && !(context.identity.roles.contains("Administrator"))
};

```

TCP flow

When a policy is evaluated, Verified Access includes data about the current TCP flow in the Cedar context under the `context.tcp_flow` key.

```

{
    "title": "TCP flow data included by Verified Access",
    "type": "object",
    "properties": {
        "destination_ip": {
            "type": "string",
            "description": "The IP address of the target",
            "example": "192.100.1.3"
        },
        "destination_port": {
            "type": "string",
            "description": "The target port",
            "example": 22
        },
        "client_ip": {
            "type": "string",
            "description": "The IP address connecting to the endpoint",

```

```

        "example": "172.154.16.9"
    }
}

```

AWS IAM Identity Center context for Verified Access trust data

When a policy is evaluated, if you define AWS IAM Identity Center as a trust provider, AWS Verified Access includes the trust data in the Cedar context under the key you specify as “Policy Reference Name” on the trust provider configuration. You can write a policy that evaluates against the trust data if you choose.

Note

The context key for your trust provider comes from the policy reference name that you configure when you create the trust provider. For example, if you configure the policy reference name as “idp123”, the context key will be “context.idp123”. Check that you are using the correct context key when you create the policy.

The following [JSON schema](#) shows which data is included in the evaluation.

```

{
  "title": "AWS IAM Identity Center context specification",
  "type": "object",
  "properties": {
    "user": {
      "type": "object",
      "properties": {
        "user_id": {
          "type": "string",
          "description": "a unique user id generated by AWS IdC"
        },
        "user_name": {
          "type": "string",
          "description": "username provided in the directory"
        },
        "email": {
          "type": "object",
          "properties": {

```

```

        "address": {
            "type": "email",
            "description": "email address associated with the user"
        },
        "verified": {
            "type": "boolean",
            "description": "whether the email address has been verified by AWS IdC"
        }
    }
}
},
"groups": {
    "type": "object",
    "description": "A list of groups the user is a member of",
    "patternProperties": {
        "^[a-zA-Z0-9]{8}-[a-zA-Z0-9]{4}-[a-zA-Z0-9]{4}-[a-zA-Z0-9]{4}-[a-zA-Z0-9]{12}$": {
            "type": "object",
            "description": "The Group ID of the group",
            "properties": {
                "group_name": {
                    "type": "string",
                    "description": "The customer-provided name of the group"
                }
            }
        }
    }
}
}
}
}

```

The following is an example of a policy that evaluates against the trust data provided by AWS IAM Identity Center.

```

permit(principal, action, resource) when {
    context.idc.user.email.verified == true
    // User is in the "sales" group with specific ID
    && context.idc.groups has "c242c5b0-6081-1845-6fa8-6e0d9513c107"
};

```

Note

As group names can be changed, IAM Identity Center refers to groups using their group ID. This helps avoid breaking a policy statement when changing the name of a group.

Third-party trust provider context for Verified Access trust data

This section describes the trust data provided to AWS Verified Access by third-party trust providers.

Note

The context key for your trust provider comes from the policy reference name that you configure when you create the trust provider. For example, if you configure the policy reference name as "idp123", the context key will be "context.idp123". Ensure you are using the correct context key when you create the policy.

Contents

- [Browser extension](#)
- [Jamf](#)
- [CrowdStrike](#)
- [JumpCloud](#)

Browser extension

If you plan to incorporate device trust context into your access policies, then you will need either the AWS Verified Access browser extension, or another partner's browser extension. Verified Access currently supports Google Chrome and Mozilla Firefox browsers.

We currently support three device trust providers: Jamf (which supports macOS devices), CrowdStrike (which supports Windows 11 and Windows 10 devices), and JumpCloud (which supports both Windows and MacOS).

- If you're using **Jamf** trust data in your policies, your users must download and install the AWS Verified Access browser extension from the [Chrome web store](#) or [Firefox Add-on site](#) on their devices.

- If you are using **CrowdStrike** trust data in your policies, first your users need to install the [AWS Verified Access Native Messaging Host](#) (direct download link). This component is required to get the trust data from the CrowdStrike agent running on users' devices. Then, after installing this component, users must install the AWS Verified Access browser extension from the [Chrome web store](#) or [Firefox Add-on site](#) on their devices.
- If you're using **JumpCloud**, your users must have the JumpCloud browser extension from the [Chrome web store](#) or [Firefox Add-on site](#) installed on their devices.

Jamf

Jamf is a third-party trust provider. When a policy is evaluated, if you define Jamf as a trust provider, Verified Access includes the trust data in the Cedar context under the key you specify as "Policy Reference Name" on the trust provider configuration. You can write a policy that evaluates against the trust data if you choose. The following [JSON schema](#) shows which data is included in the evaluation.

For more information about using Jamf with Verified Access, see [Integrating AWS Verified Access with Jamf Device Identity](#) on the Jamf website.

```
{
  "title": "Jamf device data specification",
  "type": "object",
  "properties": {
    "iss": {
      "type": "string",
      "description": "\"Issuer\" - the Jamf customer ID"
    },
    "iat": {
      "type": "integer",
      "description": "\"Issued at Time\" - a unixtime (seconds since epoch) value of when the device information data was generated"
    },
    "exp": {
      "type": "integer",
      "description": "\"Expiration\" - a unixtime (seconds since epoch) value for when this device information is no longer valid"
    },
    "sub": {
      "type": "string",
```

```

        "description": "\"Subject\" - either the hardware UID or a value generated
based on device location"
    },
    "groups": {
        "type": "array",
        "description": "Group IDs from UEM connector sync",
        "items": {
            "type": "string"
        }
    },
    "risk": {
        "type": "string",
        "enum": [
            "HIGH",
            "MEDIUM",
            "LOW",
            "SECURE",
            "NOT_APPLICABLE"
        ],
        "description": "a Jamf-reported level of risk associated with the device."
    },
    "osv": {
        "type": "string",
        "description": "The version of the OS that is currently running, in Apple
version number format (https://support.apple.com/en-us/HT201260)"
    }
}

```

The following is an example of a policy that evaluates against the trust data provided by Jamf.

```

permit(principal, action, resource) when {
    context.jamf.risk == "LOW"
};

```

Cedar provides a useful `.contains()` function to help with enums like Jamf's risk score.

```

permit(principal, action, resource) when {
    ["LOW", "SECURE"].contains(context.jamf.risk)
};

```


CrowdStrike

CrowdStrike is a third-party trust provider. When a policy is evaluated, if you define CrowdStrike as a trust provider, Verified Access includes the trust data in the Cedar context under the key you specify as “Policy Reference Name” on the trust provider configuration. You can write a policy that evaluates against the trust data if you choose. The following [JSON schema](#) shows which data is included in the evaluation.

For more information about using CrowdStrike with Verified Access, see [Securing private applications with CrowdStrike and AWS Verified Access](#) on the GitHub website.

```
{
  "title": "CrowdStrike device data specification",
  "type": "object",
  "properties": {
    "assessment": {
      "type": "object",
      "description": "Data about CrowdStrike's assessment of the device",
      "properties": {
        "overall": {
          "type": "integer",
          "description": "A single metric, between 1-100, that accounts as a weighted average of the OS and and Sensor Config scores"
        },
        "os": {
          "type": "integer",
          "description": "A single metric, between 1-100, that accounts for the OS-specific settings monitored on the host"
        },
        "sensor_config": {
          "type": "integer",
          "description": "A single metric, between 1-100, that accounts for the different sensor policies monitored on the host"
        },
        "version": {
          "type": "string",
          "description": "The version of the scoring algorithm being used"
        }
      }
    },
    "cid": {
      "type": "string",

```

```

    "description": "Customer ID (CID) unique to the customer's environment"
  },
  "exp": {
    "type": "integer",
    "description": "unixtime, The expiration time of the token"
  },
  "iat": {
    "type": "integer",
    "description": "unixtime, The issued time of the token"
  },
  "jwk_url": {
    "type": "string",
    "description": "URL that details the JWT signing"
  },
  "platform": {
    "type": "string",
    "enum": ["Windows 10", "Windows 11", "macOS"],
    "description": "Operating system of the endpoint"
  },
  "serial_number": {
    "type": "string",
    "description": "The serial number of the device derived by unique system
information"
  },
  "sub": {
    "type": "string",
    "description": "Unique CrowdStrike Agent ID (AID) of machine"
  },
  "typ": {
    "type": "string",
    "enum": ["crowdstrike-zta+jwt"],
    "description": "Generic name for this JWT media. Client MUST reject any other
type"
  }
}
}

```

The following is an example of a policy that evaluates against the trust data provided by CrowdStrike.

```

permit(principal, action, resource) when {
  context.crowdstrike.assessment.overall > 50
};

```

JumpCloud

JumpCloud is a third-party trust provider. When a policy is evaluated, if you define JumpCloud as a trust provider, Verified Access includes the trust data in the Cedar context under the key you specify as “Policy Reference Name” on the trust provider configuration. You can write a policy that evaluates against the trust data if you choose. The following [JSON schema](#) shows which data is included in the evaluation.

For more information about using JumpCloud with AWS Verified Access, see [Integrating JumpCloud and AWS Verified Access](#) on the JumpCloud website.

```
{
  "title": "JumpCloud device data specification",
  "type": "object",
  "properties": {
    "device": {
      "type": "object",
      "description": "Properties of the device",
      "properties": {
        "is_managed": {
          "type": "boolean",
          "description": "Boolean to indicate if the device is under management"
        }
      }
    },
    "exp": {
      "type": "integer",
      "description": "Expiration. Unixtime of the token's expiration."
    },
    "durt_id": {
      "type": "string",
      "description": "Device User Refresh Token ID. Unique ID that represents the device + user."
    },
    "iat": {
      "type": "integer",
      "description": "Issued At. Unixtime of the token's issuance."
    },
    "iss": {
      "type": "string",
      "description": "Issuer. This will be 'go.jumpcloud.com'"
    }
  },
}
```

```
"org_id": {
  "type": "string",
  "description": "The JumpCloud Organization ID"
},
"sub": {
  "type": "string",
  "description": "Subject. The managed JumpCloud user ID on the device."
},
"system": {
  "type": "string",
  "description": "The JumpCloud system ID"
}
}
```

The following is an example of a policy that evaluates against the trust context provided by JumpCloud.

```
permit(principal, action, resource) when {
  context.jumpcloud.org_id == 'Unique_organization_identifier'
};
```

User claims passing and signature verification in Verified Access

After an AWS Verified Access instance authenticates a user successfully, it sends the user claims received from the IdP to the Verified Access endpoint. The user claims are signed so that applications can verify the signatures and also verify that the claims were sent by Verified Access. During this process, the following HTTP header is added:

`x-amzn-ava-user-context`

This header contains the user claims in JSON web token (JWT) format. The JWT format includes a header, payload, and signature that are base64 URL encoded. Verified Access uses ES384 (ECDSA signature algorithm using SHA-384 hash algorithm) to generate the JWT signature.

Applications can use these claims for personalization or other user specific experiences. Application developers should educate themselves regarding the level of uniqueness and verification of each claim provided by the identity provider before use. In general, the sub claim is the best way to identify a given user.

Contents

- [Example: Signed JWT for OIDC user claims](#)
- [Example: Signed JWT for IAM Identity Center user claims](#)
- [Public keys](#)
- [Example: Retrieving and decoding JWT](#)

Example: Signed JWT for OIDC user claims

The following examples demonstrate what the header and payload for OIDC user claims will look like in the JWT format.

Example header:

```
{
  "alg": "ES384",
  "kid": "12345678-1234-1234-1234-123456789012",
  "signer": "arn:aws:ec2:us-east-1:123456789012:verified-access-instance/vai-abc123xzy321a2b3c",
  "iss": "OIDC Issuer URL",
  "exp": "expiration" (120 secs)
}
```

Example payload:

```
{
  "sub": "xyzsubject",
  "email": "xxx@amazon.com",
  "email_verified": true,
  "groups": [
    "Engineering",
    "finance"
  ],
  "additional_user_context": {
    "aud": "xxx",
    "exp": 10000000000,
    "groups": [
      "group-id-1",
      "group-id-2"
    ],
    "iat": 10000000000,
    "iss": "https://oidc-tp.com/",
  }
}
```

```
    "sub": "xyzsubject",
    "ver": "1.0"
  }
}
```

Example: Signed JWT for IAM Identity Center user claims

The following examples demonstrate what the header and payload for IAM Identity Center user claims will look like in the JWT format.

Note

For IAM Identity Center, only user information will be included in the claims.

Example header:

```
{
  "alg": "ES384",
  "kid": "12345678-1234-1234-1234-123456789012",
  "signer": "arn:aws:ec2:us-east-1:123456789012:verified-access-instance/vai-
abc123xzy321a2b3c",
  "iss": "arn:aws:ec2:us-east-1:123456789012:verified-access-trust-provider/vatp-
abc123xzy321a2b3c",
  "exp": "expiration" (120 secs)
}
```

Example payload:

```
{
  "user": {
    "user_id": "f478d4c8-a001-7064-6ea6-12423523",
    "user_name": "test-123",
    "email": {
      "address": "test@amazon.com",
      "verified": false
    }
  }
}
```

Public keys

Because Verified Access instances do not encrypt user claims, we recommend that you configure Verified Access endpoints to use HTTPS. If you configure your Verified Access endpoint to use HTTP, be sure to restrict the traffic to the endpoint using security groups.

To ensure security, you must verify the signature before doing any authorization based on the claims, and validate that the `signer` field in the JWT header contains the expected Verified Access instance ARN.

To get the public key, get the key ID from the JWT header and use it to look up the public key from the endpoint.

The endpoint for each AWS Region is as follows:

`https://public-keys.prod.verified-access.<region>.amazonaws.com/<key-id>`

Example: Retrieving and decoding JWT

The following code example shows how to get the key ID, public key, and payload in Python 3.9.

```
import jwt
import requests
import base64
import json

# Step 1: Validate the signer
expected_verified_access_instance_arn = 'arn:aws:ec2:region-code:account-id:verified-
access-instance/verified-access-instance-id'

encoded_jwt = headers.dict['x-amzn-ava-user-context']
jwt_headers = encoded_jwt.split('.')[0]
decoded_jwt_headers = base64.b64decode(jwt_headers)
decoded_jwt_headers = decoded_jwt_headers.decode("utf-8")
decoded_json = json.loads(decoded_jwt_headers)
received_verified_access_instance_arn = decoded_json['signer']

assert expected_verified_access_instance_arn == received_verified_access_instance_arn,
    "Invalid Signer"

# Step 2: Get the key id from JWT headers (the kid field)
kid = decoded_json['kid']
```

```
# Step 3: Get the public key from regional endpoint
url = 'https://public-keys.prod.verified-access.' + region + '.amazonaws.com/' + kid
req = requests.get(url)
pub_key = req.text

# Step 4: Get the payload
payload = jwt.decode(encoded_jwt, pub_key, algorithms=['ES384'])
```


Verified Access policies

AWS Verified Access policies allow you to define rules for accessing your applications hosted in AWS. They are written in Cedar, an AWS policy language. Using Cedar, you can create policies that are evaluated against the trust data sent from the identity or device-based trust providers that you configure to use with Verified Access.

For more detailed information about the Cedar policy language, see the [Cedar Reference Guide](#).

When you [create a Verified Access group](#) or [create a Verified Access endpoint](#), you have the option to define the Verified Access policy. You can create a group or endpoint without defining the Verified Access policy, but all access requests will be blocked until you define a policy. Alternatively, you can add or change a policy on an existing Verified Access group or endpoint after it has been created.

Contents

- [Verified Access policy statement structure](#)
- [Built-in operators for Verified Access policies](#)
- [Verified Access policy evaluation](#)
- [Verified Access policy logic short-circuiting](#)
- [Verified Access example policies](#)
- [Verified Access policy assistant](#)

Verified Access policy statement structure

The following table shows the structure of a Verified Access policy.

Component	Syntax
effect	permit forbid
scope	(principal, action, resource)
condition clause	when {

Component	Syntax
	<pre>context.<i>policy-reference-name</i> .<i>attribute-name</i> };</pre>

Policy components

A Verified Access policy contains the following components:

- **Effect** – Either permit (allow) or forbid (deny) access.
- **Scope** – The principals, actions, and resources to which the effect applies. You can leave the scope in Cedar undefined by not identifying specific principals, actions, or resources. In this case, the policy applies to all possible principals, actions, and resources.
- **Condition clause** – The context in which the effect applies.

Important

For Verified Access, policies are fully expressed by referring to trust data in the condition clause. **The policy scope must always be kept undefined.** You can then specify access using identity and device trust context in the condition clause.

Comments

You can include comments in your AWS Verified Access policies. Comments are defined as a line starting with `//` and ending with a newline character.

The following example shows comments in a policy.

```
// grants access to users in a specific domain using trusted devices
permit(principal, action, resource)
when {
  // the user's email address is in the @example.com domain
  context.idc.user.email.address.contains("@example.com")
  // Jamf thinks the user's computer is low risk or secure.
  && ["LOW", "SECURE"].contains(context.jamf.risk)
};
```

Multiple clauses

You can use more than one condition clause in a policy statement using the `&&` operator.

```
permit(principal, action, resource)
when{
  context.policy-reference-name.attribute1 &&
  context.policy-reference-name.attribute2
};
```

For additional examples, see [Verified Access example policies](#).

Reserved characters

The following example shows how to write a policy if a context property uses a `:` (semicolon), which is a reserved character in the policy language.

```
permit(principal, action, resource)
when {
  context.policy-reference-name["namespace:groups"].contains("finance")
};
```

Built-in operators for Verified Access policies

When creating the context of an AWS Verified Access policy using various conditions, as discussed in [Verified Access policy statement structure](#), you can use the `&&` operator to add additional conditions. There are also many other built-in operators that you can use to add additional expressive power to your policy conditions. The following table contains all the built-in operators for reference.

Operator	Types and overloads	Description
!	Boolean → Boolean	Logical not.
==	any → any	Equality. Works on arguments of any type, even if the types don't match. Values of different types are never equal to each other.

Operator	Types and overloads	Description
<code>!=</code>	<code>any → any</code>	Inequality; the exact inverse of equality (see above).
<code><</code>	<code>(long, long) → Boolean</code>	Long integer less-than.
<code><=</code>	<code>(long, long) → Boolean</code>	Long integer less-than-or-equal-to.
<code>></code>	<code>(long, long) → Boolean</code>	Long integer greater-than.
<code>>=</code>	<code>(long, long) → Boolean</code>	Long integer greater-than-or-equal-to.
<code>in</code>	<code>(entity, entity) → Boolean</code>	Hierarchy membership (reflexive: <code>A in A</code> is always true).
	<code>(entity, set(entity)) → Boolean</code>	Hierarchy membership: <code>A in [B, C, ...]</code> is true if <code>(A and B) (A in C) ...</code> error if the set contains a non-entity.
<code>&&</code>	<code>(Boolean, Boolean) → Boolean</code>	Logical and (short-circuiting).
<code> </code>	<code>(Boolean, Boolean) → Boolean</code>	Logical or (short-circuiting).
<code>.exists()</code>	<code>entity → Boolean</code>	Entity existence.
<code>has</code>	<code>(entity, attribute) → Boolean</code>	Infix operator. <code>e has f</code> tests if the record or entity <code>e</code> has a binding for the attribute <code>f</code> . Returns <code>false</code> if <code>e</code> does not exist or if <code>e</code> does exist but doesn't have the attribute <code>f</code> . Attributes can be expressed as identifiers or string literals.

Operator	Types and overloads	Description
like	(string, string) → Boolean	Infix operator. <code>t like p</code> checks if the text <code>t</code> matches the pattern <code>p</code> , which may include wildcard characters <code>*</code> that match 0 or more of any character. In order to match a literal star character in <code>t</code> , you can use the special escaped character sequence <code>*</code> in <code>p</code> .
.contains()	(set, any) → Boolean	Set membership (is <code>B</code> an element of <code>A</code>).
.containsAll()	(set, set) → Boolean	Tests if set <code>A</code> contains all of the elements in set <code>B</code> .
.containsAny()	(set, set) → Boolean	Tests if set <code>A</code> contains any of the elements in set <code>B</code> .

Verified Access policy evaluation

A policy document is a set of one or more policy statements (permit or forbid statements). The policy applies if the conditional clause (the when statement) is true. In order for a policy document to allow access, at least one permit policy in the document must apply and no forbid policies can apply. If no permit policies apply and/or one or more forbid policies apply, then the policy document denies access. If you have defined policy documents for both the Verified Access group and the Verified Access endpoint, both documents must allow access. If you have not defined a policy document for the Verified Access endpoint, only the Verified Access group policy needs access.

AWS Verified Access validates the syntax when you create the policy, but it does not validate the data you put in the conditional clause.

Verified Access policy logic short-circuiting

You might want to write an AWS Verified Access policy that evaluates data that may or may not be present in a given context. If you reference data in a context that does not exist, Cedar will produce an error and evaluate the policy to deny access, regardless of your intent. For example, this would result in a deny, as `fake_provider` and `bogus_key` do not exist in this context.

```
permit(principal, action, resource) when {  
    context.fake_provider.bogus_key > 42  
};
```

To avoid this situation, you can check to see if a key is present by using the `has` operator. If the `has` operator returns false, further evaluation of the chained statement halts, and Cedar does not produce an error attempting to reference an item that does not exist.

```
permit(principal, action, resource) when {  
    context.identity.user has "some_key" && context.identity.user.some_key > 42  
};
```

This is most useful when specifying a policy that references two different trust providers.

```
permit(principal, action, resource) when {  
    // user is in an allowed group  
    context.aws_idc.groups has "c242c5b0-6081-1845-6fa8-6e0d9513c107"  
    &&(  
        (  
            // if CrowdStrike data is present,  
            // permit if CrowdStrike's overall assessment is over 50  
            context has "crowdstrike" && context.crowdstrike.assessment.overall > 50  
        )  
        ||  
        (  
            // if Jamf data is present,  
            // permit if Jamf's risk score is acceptable  
            context has "jamf" && ["LOW", "NOT_APPLICABLE", "MEDIUM",  
"SECURE"].contains(context.jamf.risk)  
        )  
    )  
};
```

Verified Access example policies

You can use Verified Access policies to grant access to your applications to specific users and devices.

Example policies

- [Example 1: Grant access to a group in IAM Identity Center](#)
- [Example 2: Grant access to a group in a third-party provider](#)
- [Example 3: Grant access using CrowdStrike](#)
- [Example 4: Allow or deny a specific IP address](#)

Example 1: Grant access to a group in IAM Identity Center

When using AWS IAM Identity Center, it is better to refer to groups by using their IDs. This helps to avoid breaking a policy statement if you change the name of the group.

The following example policy allows access only to users in the specified group with a verified email address. The group ID is c242c5b0-6081-1845-6fa8-6e0d9513c107.

```
permit(principal,action,resource)
when {
    context.policy-reference-name.groups has "c242c5b0-6081-1845-6fa8-6e0d9513c107"
    && context.policy-reference-name.user.email.verified == true
};
```

The following example policy allows access only when the user is in the specified group, the user has a verified email address, and the Jamf device risk score is LOW.

```
permit(principal,action,resource)
when {
    context.policy-reference-name.groups has "c242c5b0-6081-1845-6fa8-6e0d9513c107"
    && context.policy-reference-name.user.email.verified == true
    && context.jamf.risk == "LOW"
};
```

For more information about the trust data, see [the section called “AWS IAM Identity Center context”](#).

Example 2: Grant access to a group in a third-party provider

The following example policy allows access only when the user is in the specified group, the user has a verified email address, and the Jamf device risk score is LOW. The name of the group is "finance".

```
permit(principal, action, resource)
when {
    context.policy-reference-name.groups.contains("finance")
    && context.policy-reference-name.email_verified == true
    && context.jamf.risk == "LOW"
};
```

For more information about the trust data, see [the section called "Third-party context"](#).

Example 3: Grant access using CrowdStrike

The following example policy allows access when the overall assessment score is greater than 50.

```
permit(principal, action, resource)
when {
    context.crowd.assessment.overall > 50
};
```

Example 4: Allow or deny a specific IP address

The following example policy allows requests only from the specified IP address.

```
permit(principal, action, resource)
when {
    context.http_request.client_ip == "192.0.2.1"
};
```

The following example policy denies requests from the specified IP address.

```
forbid(principal, action, resource)
when {
    ip(context.http_request.client_ip).isInRange(ip("192.0.2.1/32"))
};
```


Verified Access policy assistant

The Verified Access policy assistant is a tool in the Verified Access console that you can use to test and develop your policies. It presents the endpoint policy, the group policy, and the trust context on one screen, where you can test and make edits to the policies.

Trust context formats vary across different trust providers, and sometimes the Verified Access administrator might not know the exact format a certain trust provider uses. That is why it can be very helpful to see the trust context, and both the group and endpoint policies in one place for testing and developing purposes.

The following sections describe the basics of using the policy editor.

Tasks

- [Step 1: Specify your resources](#)
- [Step 2: Test and edit policies](#)
- [Step 3: Review and apply changes](#)

Step 1: Specify your resources

On the first page of the policy assistant, you specify the Verified Access endpoint that you want to work with. You will also specify a user (identified by email address), and optionally, the user's name and/or a device identifier. By default, the most recent authorization decision is extracted from the Verified Access logs for the specified user. You can optionally choose the most recent allow *or* deny decision specifically.

Finally, the trust context, authorization decision, endpoint policy, and group policy are all displayed on the next screen.

To open the policy assistant and specify your resources

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access instances**, then click the **Verified Access instance ID** for the instance you want to work with.
3. Choose **Launch policy assistant**.
4. For **User email address**, enter the email address of the user.
5. For **Verified Access endpoint**, select the endpoint that you want to edit and test policies for.

6. (Optional) For **Name**, provide the name of the user.
7. (Optional) Under **Device identifier**, provide the unique device identifier.
8. (Optional) For **Authorization result**, choose the type of recent authorization result you want to use. By default, the latest authorization result will be used.
9. Choose **Next**.

Step 2: Test and edit policies

On this page you will be presented with the following information to work with:

- The trust context sent by your trust provider for the user and (optionally) the device that you specified in the previous step.
- The Cedar policy for the Verified Access endpoint specified in the previous step.
- The Cedar policy for the Verified Access group that the endpoint belongs to.

The Cedar policies for the Verified Access endpoint and group can be edited on this page, but the trust context is static. You can now use this page to view the trust context along side the Cedar policies.

Test the policies against the trust context by choosing the **Test policies** button, and the authorization result will be displayed on the screen. You can make edits to the policies and retest your changes, repeating the process as needed.

After you are satisfied with the changes made to the policies, choose **Next** to continue to the next screen of the policy assistant.

Step 3: Review and apply changes

On the final page of the policy assistant, you will see the changes you made to the policies highlighted for easy review. You can now review them a final time and choose **Apply changes** to commit the changes.

You also have the option of going back to the previous page by choosing **Previous**, or cancelling out of the policy assistant completely by choosing **Cancel**.

Connectivity Client for AWS Verified Access

AWS Verified Access provides the Connectivity Client so that you can enable connectivity between user devices and non-HTTP applications. The client securely encrypts user traffic, adds user identity information and device context, and routes it to Verified Access for policy enforcement. If the access policies allow access, the user is connected to the application. User access is continuously authorized for as long as the Connectivity Client is connected.

The client runs as a system service and is resilient against crashes. If the connection becomes unsteady, the client reestablishes the connection.

The client uses ephemeral OAuth access tokens to establish the secure tunnel. The tunnel is disconnected when the user signs out of the client.

Access and refresh tokens are stored locally on the user device, in an encrypted SQLite database.

Contents

- [Prerequisites](#)
- [Download the Connectivity Client](#)
- [Export the client configuration file](#)
- [Connect to the application](#)
- [Uninstall the client](#)
- [Best practices](#)
- [Troubleshooting](#)
- [Version history](#)

Prerequisites

Before you begin, complete the following prerequisites:

- Create a Verified Access instance with a trust provider.
- Create a TCP endpoint for your application.
- Disconnect your computer from any VPN clients to avoid routing issues.
- Enable IPv6 on your computer. For instructions, see the documentation for the operating system that is running on your computer.

- On a Windows computer, verify that [Trusted Platform Module \(TPM\)](#) is supported and install the [WebView2](#) runtime.

Download the Connectivity Client

Uninstall any previous version of the client. Download the client, verify that the installer is signed, and run the installer. Do not install the client using an unsigned installer.

- [Connectivity Client for Mac with Apple Silicon version 1.0.2](#)
- [Connectivity Client for Mac with Intel version 1.0.2](#)
- [Connectivity Client for Windows with x64 version 1.0.2](#)

Export the client configuration file

Use the following procedure to export the configuration information required by the client from your Verified Access instance.

To export the client configuration file using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access instances**.
3. Select the Verified Access instance.
4. Choose **Actions, Export client configuration file**.

To export the client configuration file using the AWS CLI

Use the [export-verified-access-instance-client-configuration](#) command. Save the output to a .json file. The file name must start with the ClientConfig- prefix.

Connect to the application

Use the following procedure to connect to an application using the client.

To connect to an application using the client

1. Deploy the client configuration files to the users' devices in the following location:

- Windows – C:\ProgramData\Connectivity Client
 - macOS – /Library/Application Support/Connectivity Client
2. Ensure that the client configuration files are owned by root (macOS) or Admin (Windows).
 3. Launch the Connectivity Client.
 4. After the Connectivity Client is loaded, the user is authenticated by the IdP.
 5. After authentication, users can access the application using the DNS name provided by Verified Access, using the client of their choice.

Uninstall the client

When you are finished using the Connectivity Client, you can uninstall it.

macOS

Version 1.0.1 and later

Navigate to /Applications/Connectivity Client and run Connectivity Client Uninstaller.app.

Version 1.0.0

Download the `connectivity_client_cleanup.sh` script for [Mac with Apple Silicon](#) or [Mac with Intel](#), set execution permissions on the script, and run the script as follows.

```
sudo ./connectivity_client_cleanup.sh
```

Windows

To uninstall the client on Windows, run the installer and choose **Remove**.

Best practices

Consider the following best practices:

- Install the latest version of the client.
- Do not install the client using an unsigned installer.

- Users should not use a configuration unless it is a trusted configuration provided by an IT admin. An untrusted configuration could redirect to a phishing page.
- Users should sign out of the client before leaving their workstations idle.
- Add the `offline_access` scope to your OIDC configuration. This allows requests for refresh tokens, which are used to obtain more access tokens without requiring the user to re-authenticate.

Troubleshooting

The following information can help you troubleshoot issues with the client.

Issues

- [When signing in, the browser doesn't open to complete authentication by the IdP](#)
- [After authentication, the client status is "not connected"](#)
- [Can't connect using a Chrome or Edge browser](#)

When signing in, the browser doesn't open to complete authentication by the IdP

Possible cause: The configuration file is missing or malformed.

Solution: Contact your system administrator and request an updated configuration file.

After authentication, the client status is "not connected"

Possible cause: Running other VPN software, such as AWS Client VPN, Cisco AnyConnect, or OpenVPN Connect.

Solution: Disconnect from any other VPN software. If you're still unable to connect, generate a diagnostic report and share it with your system administrator.

Possible cause: On Windows platforms, the client uses HTTP on port 80 for control plane communication. A firewall rule that blocks TCP port 80 prevents control plane communication.

Solution: Check Windows Firewall rules for an explicit outbound rule blocking TCP on port 80 and disable it.

Can't connect using a Chrome or Edge browser

Possible cause: When connecting to a web application using a Chrome or Edge browser, the browser fails to resolve the IPv6 domain name.

Solution: Contact [AWS Support](#).

Version history

The following table contains the version history of the client.

Version	Changes	Download	Date
1.0.2	macOS <ul style="list-style-type: none">Bug fixes and stability improvementsUI enhancements Windows <ul style="list-style-type: none">Bug fixes and stability improvementsUI enhancements	<ul style="list-style-type: none">Mac with Apple SiliconMac with IntelWindows with x64	June 9, 2025
1.0.1	macOS <ul style="list-style-type: none">Stability improvementsUninstaller application Windows <ul style="list-style-type: none">Stability improvements	<ul style="list-style-type: none">Mac with Apple SiliconMac with IntelWindows with x64	February 5, 2025
1.0.0	Public preview	<ul style="list-style-type: none">Mac with Apple SiliconMac with IntelWindows with x64	December 1, 2024

Security in Verified Access

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

Contents

- [Data protection in Verified Access](#)
- [Identity and access management for Verified Access](#)
- [Compliance validation for Verified Access](#)
- [Resilience in Verified Access](#)

Data protection in Verified Access

The AWS [shared responsibility model](#) applies to data protection in AWS Verified Access. 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 Verified Access 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

Verified Access encrypts all data in transit from end users to Verified Access endpoints over the Internet using Transport Layer Security (TLS) 1.2 or later.

Inter-network traffic privacy

You can configure Verified Access to restrict access to specific resources in your VPC. For user-based authentication you can also restrict access to portions of your network, based on the user group that accesses the endpoints. For more information, see [Verified Access policies](#).

Data encryption at rest for AWS Verified Access

AWS Verified Access encrypts data at rest by default, using AWS owned KMS keys. When encryption of data at rest happens by default, it helps reduce the operational overhead and complexity that are involved in protecting sensitive data. At the same time, it enables you to build secure applications that meet strict encryption compliance and regulatory requirements. The following sections provide the details of how Verified Access uses KMS keys for data encryption at rest.

Contents

- [Verified Access and KMS keys](#)
- [Personally identifiable information](#)
- [How AWS Verified Access uses grants in AWS KMS](#)
- [Using customer managed keys with Verified Access](#)
- [Specifying a customer managed key for Verified Access resources](#)
- [AWS Verified Access encryption context](#)
- [Monitoring your encryption keys for AWS Verified Access](#)

Verified Access and KMS keys

AWS owned keys

Verified Access uses KMS keys to automatically encrypt personally identifiable information (PII). This happens by default, and you can't yourself view, manage, use, or audit the use of the AWS owned keys. However, you don't have to take any action or change any programs to protect the keys that encrypt your data. For more information, see [AWS owned keys](#) in the *AWS Key Management Service Developer Guide*.


While you can't disable this layer of encryption or select an alternate encryption type, you can add a second layer of encryption over the existing AWS owned encryption keys by choosing a customer managed key when you create your Verified Access resources.

Customer managed keys

Verified Access supports the use of symmetric customer managed keys that you create and manage, to add a second layer of encryption over the existing default encryption. Because you have full control of this layer of encryption, you can perform such tasks as:

- Establishing and maintaining key policies
- Establishing and maintaining IAM policies and grants
- Enabling and disabling key policies
- Rotating key cryptographic material
- Adding tags
- Creating key aliases
- Scheduling keys for deletion

For more information, see [Customer managed keys](#) in the *AWS Key Management Service Developer Guide*.

 **Note**

Verified Access automatically enables encryption at rest using AWS owned keys to protect personally identifiable data at no charge.
However, AWS KMS charges will apply when you use a customer managed key. For more information about pricing, see the [AWS Key Management Service pricing](#).

Personally identifiable information

The following table summarizes the personally identifiable information (PII) that Verified Access uses, and how it is encrypted.

Data type	AWS owned key encryption	Customer managed key encryption (Optional)
Trust provider (user-type)	Enabled	Enabled
User-type trust providers contain OIDC options such		

Data type	AWS owned key encryption	Customer managed key encryption (Optional)
as AuthorizationEndpoint, UserInfoEndpoint, ClientId, ClientSecret, and so on, which are considered PII.		
Trust provider (device-type) Device-type trust providers contain a TenantId, which is considered PII.	Enabled	Enabled
Group policy Provided during creation or modification of Verified Access group. Contains rules for authorizing access requests. Might contain PII such as username and email address, and so on.	Enabled	Enabled
Endpoint policy Provided during creation or modification of Verified Access endpoint. Contains rules for authorizing access requests. Might contain PII such as username and email address, and so on.	Enabled	Enabled

How AWS Verified Access uses grants in AWS KMS

Verified Access requires a [grant](#) to use your customer managed key.

When you create Verified Access resources encrypted with a customer managed key, Verified Access creates a grant on your behalf by sending a [CreateGrant](#) request to AWS KMS. Grants in AWS KMS are used to give Verified Access the access to a customer managed key in your account.

Verified Access requires the grant to use your customer managed key for the following internal operations:

- Send [Decrypt](#) requests to AWS KMS to decrypt the encrypted data keys so that they can be used to decrypt your data.
- Send [RetireGrant](#) requests to AWS KMS to delete a grant.

You can revoke access to the grant, or remove the service's access to the customer managed key at any time. If you do, Verified Access won't be able to access any of the data that's encrypted by the customer managed key, which affects operations that are dependent on that data.

Using customer managed keys with Verified Access

You can create a symmetric customer managed key by using the AWS Management Console, or the AWS KMS APIs. Follow the steps for [Creating a symmetric encryption key](#) in the *AWS Key Management Service Developer Guide*.

Key policies

Key policies control access to your customer managed key. Every customer managed key must have exactly one key policy, which contains statements that determine who can use the key and how they can use it. When you create your customer managed key, you can specify a key policy. For more information, see [Key policies](#) in the *AWS Key Management Service Developer Guide*.

To use your customer managed key with your Verified Access resources, the following API operations must be permitted in the key policy:

- [kms:CreateGrant](#) – Adds a grant to a customer managed key. Grants control access to a specified KMS key, which allows access to [grant operations](#) Verified Access requires. For more information, see [Grants](#), in the *AWS Key Management Service Developer Guide*.

This allows Verified Access to do the following:

- Call `GenerateDataKeyWithoutPlainText` to generate an encrypted data key and store it, because the data key isn't immediately used to encrypt.
- Call `Decrypt` to use the stored encrypted data key to access encrypted data.

- Set up a retiring principal to allow the service to RetireGrant.
- [kms:DescribeKey](#) – Provides the customer managed key details to allow Verified Access to validate the key.
- [kms:GenerateDataKey](#) – Allows Verified Access to use key for encrypting data.
- [kms:Decrypt](#) – Allow Verified Access to decrypt the encrypted data keys.

The following is an example key policy you can use for Verified Access.

```
"Statement" : [
  {
    "Sid" : "Allow access to principals authorized to use Verified Access",
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "*"
    },
    "Action" : [
      "kms:DescribeKey",
      "kms:CreateGrant",
      "kms:GenerateDataKey",
      "kms:Decrypt"
    ],
    "Resource" : "*",
    "Condition" : {
      "StringEquals" : {
        "kms:ViaService" : "verified-access.region.amazonaws.com",
        "kms:CallerAccount" : "111122223333"
      }
    }
  },
  {
    "Sid": "Allow access for key administrators",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::111122223333:root"
    },
    "Action" : [
      "kms:*"
    ],
    "Resource": "arn:aws:kms:region:111122223333:key/key_ID"
  },
  {
    "Sid" : "Allow read-only access to key metadata to the account",
```

```
"Effect" : "Allow",
"Principal" : {
  "AWS" : "arn:aws:iam::111122223333:root"
},
"Action" : [
  "kms:Describe*",
  "kms:Get*",
  "kms:List*",
  "kms:RevokeGrant"
],
"Resource" : "*"
}
```

For more information, see [Creating a key policy](#) and [troubleshooting key access](#) in the *AWS Key Management Service Developer Guide*.

Specifying a customer managed key for Verified Access resources

You can specify a customer managed key to provide a second layer encryption for the following resources:

- [Verified Access group](#)
- [Verified Access endpoint](#)
- [Verified Access trust provider](#)

When you create any of these resources using the AWS Management Console, you can specify a customer managed key in the **Additional encryption -- optional** section. During the process, select the **Customize encryption settings (advanced)** check box, then enter the AWS KMS key ID you want to use. This can also be done when modifying an existing resource, or by using the AWS CLI.

Note

If the customer managed key used to add additional encryption to any of the above resources is lost, the configuration values for the resources will no longer be accessible. The resources can be modified however, by using the AWS Management Console or AWS CLI, to apply a new customer managed key and reset the configuration values.

AWS Verified Access encryption context

An [encryption context](#) is an optional set of key-value pairs that contain additional contextual information about the data. AWS KMS uses the encryption context as additional authenticated data to support authenticated encryption. When you include an encryption context in a request to encrypt data, AWS KMS binds the encryption context to the encrypted data. To decrypt data, you include the same encryption context in the request.

AWS Verified Access encryption context

Verified Access uses the same encryption context in all AWS KMS cryptographic operations, where the key is `aws:verified-access:arn` and the value is the resource Amazon Resource Name (ARN). Below are the encryption contexts for Verified Access resources.

Verified Access trust provider

```
"encryptionContext": {
  "aws:verified-access:arn":
    "arn:aws:ec2:region:111122223333:VerifiedAccessTrustProviderId"
}
```

Verified Access group

```
"encryptionContext": {
  "aws:verified-access:arn":
    "arn:aws:ec2:region:111122223333:VerifiedAccessGroupId"
}
```

Verified Access endpoint

```
"encryptionContext": {
  "aws:verified-access:arn":
    "arn:aws:ec2:region:111122223333:VerifiedAccessEndpointId"
}
```

Monitoring your encryption keys for AWS Verified Access

When you use a customer managed KMS key with your AWS Verified Access resources, you can use [AWS CloudTrail](#) to track requests that Verified Access sends to AWS KMS.

The following examples are AWS CloudTrail events for CreateGrant, RetireGrant, Decrypt, DescribeKey, and GenerateDataKey, which monitor KMS operations called by Verified Access to access data that's encrypted by your customer managed KMS key:

CreateGrant

When you use a customer managed key to encrypt your resources, Verified Access sends a CreateGrant request on your behalf to access the key in your AWS account. The grant that Verified Access creates is specific to the resource that's associated with the customer managed key.

The following example event records the CreateGrant operation:

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/Admin",
        "accountId": "111122223333",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2023-09-11T16:27:12Z",
        "mfaAuthenticated": "false"
      }
    },
    "invokedBy": "verified-access.amazonaws.com"
  },
  "eventTime": "2023-09-11T16:41:42Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "CreateGrant",
  "awsRegion": "ca-central-1",
  "sourceIPAddress": "verified-access.amazonaws.com",
  "userAgent": "verified-access.amazonaws.com",
```

```

    "requestParameters": {
      "operations": [
        "Decrypt",
        "RetireGrant",
        "GenerateDataKey"
      ],
      "keyId": "arn:aws:kms:ca-central-1:111122223333:key/5ed79e7f-88c9-420c-ae1a-61ee87104dae",
      "constraints": {
        "encryptionContextSubset": {
          "aws:verified-access:arn": "arn:aws:ec2:ca-central-1:111122223333:verified-access-trust-provider/vatp-0e54f581e2e5c97a2"
        }
      },
      "granteePrincipal": "verified-access.ca-central-1.amazonaws.com",
      "retiringPrincipal": "verified-access.ca-central-1.amazonaws.com"
    },
    "responseElements": {
      "grantId":
        "e5a050fff9893ba1c43f83fddf61e5f9988f579beaadd6d4ad6d1df07df6048f",
      "keyId": "arn:aws:kms:ca-central-1:111122223333:key/5ed79e7f-88c9-420c-ae1a-61ee87104dae"
    },
    "requestID": "0faa837e-5c69-4189-9736-3957278e6444",
    "eventID": "1b6dd8b8-cbee-4a83-9b9d-d95fa5f6fd08",
    "readOnly": false,
    "resources": [
      {
        "accountId": "AWS Internal",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:ca-central-1:111122223333:key/5ed79e7f-88c9-420c-ae1a-61ee87104dae"
      }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
  }

```

RetireGrant

Verified Access uses the `RetireGrant` operation to remove a grant when you delete a resource.

The following example event records the RetireGrant operation:

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/Admin",
        "accountId": "111122223333",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2023-09-11T16:42:33Z",
        "mfaAuthenticated": "false"
      }
    },
    "invokedBy": "verified-access.amazonaws.com"
  },
  "eventTime": "2023-09-11T16:47:53Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "RetireGrant",
  "awsRegion": "ca-central-1",
  "sourceIPAddress": "verified-access.amazonaws.com",
  "userAgent": "verified-access.amazonaws.com",
  "requestParameters": null,
  "responseElements": {
    "keyId": "arn:aws:kms:ca-central-1:111122223333:key/5ed79e7f-88c9-420c-ae1a-61ee87104dae"
  },
  "additionalEventData": {
    "grantId": "b35e66f9bacb266cec214fcaa353c9cf750785e28773e61ba6f434d8c5c7632f"
  },
  "requestID": "7d4a31c2-d426-434b-8f86-336532a70462",
  "eventID": "17edc343-f25b-43d4-bbfff-150d8ffff4cf8",
  "readOnly": false,
}
```

```
"resources": [
  {
    "accountId": "AWS Internal",
    "type": "AWS::KMS::Key",
    "ARN": "arn:aws:kms:ca-central-1:111122223333:key/5ed79e7f-88c9-420c-ae1a-61ee87104dae"
  }
],
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111122223333",
"eventCategory": "Management"
}
```

Decrypt

Verified Access calls the Decrypt operation to use the stored encrypted data key to access the encrypted data.

The following example event records the Decrypt operation:

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/Admin",
        "accountId": "111122223333",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2023-09-11T17:19:33Z",
        "mfaAuthenticated": "false"
      }
    },
  },
  "invokedBy": "verified-access.amazonaws.com"
```

```

    },
    "eventTime": "2023-09-11T17:47:05Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "ca-central-1",
    "sourceIPAddress": "verified-access.amazonaws.com",
    "userAgent": "verified-access.amazonaws.com",
    "requestParameters": {
      "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
      "keyId": "arn:aws:kms:ca-central-1:111122223333:key/380d006e-706a-464b-99c5-68768297114e",
      "encryptionContext": {
        "aws:verified-access:arn": "arn:aws:ec2:ca-central-1:111122223333:verified-access-trust-provider/vatp-00f20a4e455e9340f",
        "aws-crypto-public-key": "AkK+vi1W/acBKv70R8p2DeUrA8EgpTffSrjBqNuc0DuBYhyZ3hlMuYYJz9x7CwQWZw=="
      }
    },
    "responseElements": null,
    "requestID": "2e920fd3-f2f6-41b2-a5e7-2c2cb6f853a9",
    "eventID": "3329e0a3-bcfb-44cf-9813-8106d6eee31d",
    "readOnly": true,
    "resources": [
      {
        "accountId": "AWS Internal",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:ca-central-1:111122223333:key/380d006e-706a-464b-99c5-68768297114e"
      }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
  }

```

DescribeKey

Verified Access uses the `DescribeKey` operation to verify whether the customer managed key that's associated with your resource exists in the account and Region.

The following example event records the `DescribeKey` operation:

```
{
```

```

"eventVersion": "1.08",
"userIdentity": {
  "type": "AssumedRole",
  "principalId": "AKIAI44QH8DHBEXAMPLE",
  "arn": "arn:aws:sts::111122223333:assumed-role/Admin/",
  "accountId": "111122223333",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
    "sessionIssuer": {
      "type": "Role",
      "principalId": "AKIAI44QH8DHBEXAMPLE",
      "arn": "arn:aws:iam::111122223333:role/Admin",
      "accountId": "111122223333",
      "userName": "Admin"
    },
    "webIdFederationData": {},
    "attributes": {
      "creationDate": "2023-09-11T17:19:33Z",
      "mfaAuthenticated": "false"
    }
  },
  "invokedBy": "verified-access.amazonaws.com"
},
"eventTime": "2023-09-11T17:46:48Z",
"eventSource": "kms.amazonaws.com",
"eventName": "DescribeKey",
"awsRegion": "ca-central-1",
"sourceIPAddress": "verified-access.amazonaws.com",
"userAgent": "verified-access.amazonaws.com",
"requestParameters": {
  "keyId": "arn:aws:kms:ca-central-1:111122223333:key/380d006e-706a-464b-99c5-68768297114e"
},
"responseElements": null,
"requestID": "5b127082-6691-48fa-bfb0-4d40e1503636",
"eventID": "ffcfc2bb-f94b-4c00-b6fb-feac77daff2a",
"readOnly": true,
"resources": [
  {
    "accountId": "AWS Internal",
    "type": "AWS::KMS::Key",
    "ARN": "arn:aws:kms:ca-central-1:111122223333:key/380d006e-706a-464b-99c5-68768297114e"
  }
]

```

```

    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
  }

```

GenerateDataKey

The following example event records the GenerateDataKey operation:

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/Admin",
        "accountId": "111122223333",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2023-09-11T17:19:33Z",
        "mfaAuthenticated": "false"
      }
    },
    "invokedBy": "verified-access.amazonaws.com"
  },
  "eventTime": "2023-09-11T17:46:49Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateDataKey",
  "awsRegion": "ca-central-1",
  "sourceIPAddress": "verified-access.amazonaws.com",
  "userAgent": "verified-access.amazonaws.com",
  "requestParameters": {
    "encryptionContext": {

```

```

    "aws:verified-access:arn": "arn:aws:ec2:ca-
central-1:111122223333:verified-access-trust-provider/vatp-00f20a4e455e9340f",
    "aws-crypto-public-key": "A/ATGxaYatPU10tM+l/mfDndkzHUmX5Hav+29I1Im
+JRBKFuXf24ulztm0IsqFQliw=="
  },
  "numberOfBytes": 32,
  "keyId": "arn:aws:kms:ca-
central-1:111122223333:key/380d006e-706a-464b-99c5-68768297114e"
},
  "responseElements": null,
  "requestID": "06535808-7cce-4ae1-ab40-e3afbf158a43",
  "eventID": "1ce79601-5a5e-412c-90b3-978925036526",
  "readOnly": true,
  "resources": [
    {
      "accountId": "AWS Internal",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:ca-
central-1:111122223333:key/380d006e-706a-464b-99c5-68768297114e"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "eventCategory": "Management"
}

```

Identity and access management for Verified Access

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 Verified Access resources. IAM is an AWS service that you can use with no additional charge.

Topics

- [Audience](#)
- [Authenticating with identities](#)
- [Managing access using policies](#)
- [How Verified Access works with IAM](#)

- [Identity-based policy examples for Verified Access](#)
- [Troubleshooting Verified Access identity and access](#)
- [Use service-linked roles for Verified Access](#)
- [AWS managed policies for Verified Access](#)

Audience

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

Service user – If you use the Verified Access service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Verified Access 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 Verified Access, see [Troubleshooting Verified Access identity and access](#).

Service administrator – If you're in charge of Verified Access resources at your company, you probably have full access to Verified Access. It's your job to determine which Verified Access 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 Verified Access, see [How Verified Access 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 Verified Access. To view example Verified Access identity-based policies that you can use in IAM, see [Identity-based policy examples for Verified Access](#).

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 Verified Access works with IAM

Before you use IAM to manage access to Verified Access, learn what IAM features are available to use with Verified Access.

IAM feature	Verified Access 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

IAM feature	Verified Access support
Temporary credentials	Yes
Principal permissions	Yes
Service roles	No
Service-linked roles	Yes

To get a high-level view of how Verified Access 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 Verified Access

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 Verified Access

To view examples of Verified Access identity-based policies, see [Identity-based policy examples for Verified Access](#).

Resource-based policies within Verified Access

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 Verified Access

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 Verified Access actions, see [Actions Defined by Amazon EC2](#) in the *Service Authorization Reference*.

Policy actions in Verified Access use the following prefix before the action:

```
ec2
```

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

```
"Action": [  
    "ec2:action1",  
    "ec2:action2"  
]
```

To view examples of Verified Access identity-based policies, see [Identity-based policy examples for Verified Access](#).

Policy resources for Verified Access

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 Verified Access resource types and their ARNs, see [Resources Defined by Amazon EC2](#) in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see [Actions Defined by Amazon EC2](#).

To view examples of Verified Access identity-based policies, see [Identity-based policy examples for Verified Access](#).

Policy condition keys for Verified Access

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 Verified Access condition keys, see [Condition Keys for Amazon EC2](#) in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see [Actions Defined by Amazon EC2](#).

To view examples of Verified Access identity-based policies, see [Identity-based policy examples for Verified Access](#).

ACLs in Verified Access

Supports ACLs: No

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

ABAC with Verified Access

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 Verified Access

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 Verified Access

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 Verified Access

Supports service roles: No

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

Service-linked roles for Verified Access

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 Verified Access service-linked roles, see [Use service-linked roles for Verified Access](#).

Identity-based policy examples for Verified Access

By default, users and roles don't have permission to create or modify Verified Access 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 Verified Access, including the format of the ARNs for each of the resource types, see [Actions, Resources, and Condition Keys for Amazon EC2](#) in the *Service Authorization Reference*.

Topics

- [Policy best practices](#)
- [Policy for creating Verified Access instances](#)
- [Allow users to view their own permissions](#)

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Verified Access 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*.

Policy for creating Verified Access instances

To create a Verified Access instance, IAM principals need to add this additional statement to their IAM policy.

```
{
  "Effect": "Allow",
  "Action": "verified-access:AllowVerifiedAccess",
  "Resource": "*"
}
```

Note

`verified-access:AllowVerifiedAccess` is an action-only virtual API. It does not support resource, tag, or condition key-based authorization. Use resource, tag, or condition key-based authorization on the `ec2:CreateVerifiedAccessInstance` API action.

Example policy for creating a Verified Access instance. In this example, 123456789012 is the AWS account number and us-east-1 is the AWS region.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:CreateVerifiedAccessInstance",
      "Resource": "arn:aws:ec2:us-east-1:123456789012:verified-access-instance/*"
    },
    {
      "Effect": "Allow",
```

```

        "Action": "verified-access:AllowVerifiedAccess",
        "Resource": "*"
    }
]
}

```

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 Verified Access identity and access

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

Issues

- [I am not authorized to perform an action in Verified Access](#)
- [I am not authorized to perform iam:PassRole](#)
- [I want to allow people outside of my AWS account to access my Verified Access resources](#)

I am not authorized to perform an action in Verified Access

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

The following example error occurs when the mateojackson IAM user tries to use the console to view details about a fictional *my-example-widget* resource but doesn't have the fictional *ec2:GetWidget* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
ec2:GetWidget on resource: my-example-widget
```

In this case, the policy for the mateojackson user must be updated to allow access to the *my-example-widget* resource by using the *ec2:GetWidget* action.

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

I am not authorized to perform iam:PassRole

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

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 Verified Access. 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 allow people outside of my AWS account to access my Verified Access 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 Verified Access supports these features, see [How Verified Access 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*.

Use service-linked roles for Verified Access

AWS Verified Access uses an IAM service-linked role, which is a type of IAM role that is linked directly to an AWS service. The service-linked roles for Verified Access are defined by Verified

Access and include all the permissions that the service requires to call other AWS services on your behalf.

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

Service-linked role permissions for Verified Access

Verified Access uses the service-linked role named **AWSServiceRoleForVPCVerifiedAccess** to provision resources in your account that are required to use the service.

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

- `verified-access.amazonaws.com`

The role permissions policy, named **AWSVPCVerifiedAccessServiceRolePolicy**, allows Verified Access to complete the following actions on the specified resources:

- Action `ec2:CreateNetworkInterface` on all subnets and security groups, as well as all network interfaces with the tag `VerifiedAccessManaged=true`
- Action `ec2:CreateTags` on all network interfaces at creation time
- Action `ec2:DeleteNetworkInterface` on all network interfaces with the tag `VerifiedAccessManaged=true`
- Action `ec2:ModifyNetworkInterfaceAttribute` on all security groups and all network interfaces with the tag `VerifiedAccessManaged=true`

You can also view the permissions for this policy in the *AWS Managed Policy Reference Guide*; see [AWSVPCVerifiedAccessServiceRolePolicy](#).

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*.

Create a service-linked role for Verified Access

You don't need to manually create a service-linked role. When you call **CreateVerifiedAccessEndpoint** in the AWS Management Console, the AWS CLI, or the AWS API, Verified Access 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. When you call **CreateVerifiedAccessEndpoint** once again, Verified Access creates the service-linked role for you again.

Edit a service-linked role for Verified Access

Verified Access does not allow you to edit the **AWSServiceRoleForVPCVerifiedAccess** 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 [Edit a service-linked role description](#) in the *IAM User Guide*.

Delete a service-linked role for Verified Access

You don't need to manually delete the **AWSServiceRoleForVPCVerifiedAccess** role. When you call **DeleteVerifiedAccessEndpoint** in the AWS Management Console, the AWS CLI, or the AWS API, Verified Access cleans up the resources and deletes the service-linked role for you.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the **AWSServiceRoleForVPCVerifiedAccess** service-linked role. For more information, see [Delete a service-linked role](#) in the *IAM User Guide*.

Supported Regions for Verified Access service-linked roles

Verified Access supports using service-linked roles in all of the AWS Regions where the service is available. For more information, see [AWS Regions and endpoints](#).

AWS managed policies for Verified Access

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: AWSVPCVerifiedAccessServiceRolePolicy

This policy is attached to a service-linked role that allows Verified Access to perform actions on your behalf. For more information, see [Use service-linked roles](#). To view the permissions for this policy, you can see [AWSVPCVerifiedAccessServiceRolePolicy](#) in the AWS Management Console, or you can view the [AWSVPCVerifiedAccessServiceRolePolicy](#) policy in the *AWS Managed Policy Reference Guide*.

Verified Access updates to AWS managed policies

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

Change	Description	Date
AWSVPCVerifiedAccessServiceRolePolicy - Policy updated	Verified Access updated its managed policy to include descriptions of all actions under the "sid" field.	November 17, 2023
AWSVPCVerifiedAccessServiceRolePolicy - Policy updated	Verified Access updated its managed policy to add security group resource to <code>ec2:CreateNetworkInterface</code> permission.	May 31, 2023

Change	Description	Date
AWSVPCVerifiedAccessServiceRolePolicy - New policy	Verified Access added a new policy to allow it to provision resources in your account that are required to use the service.	November 29, 2022
Verified Access started tracking changes	Verified Access started tracking changes for its AWS managed policies.	November 29, 2022

Compliance validation for Verified Access

AWS Verified Access can be configured to support Federal Information Processing Standards (FIPS) compliance. For more info and details on setting up FIPS compliance for Verified Access, go to [FIPS compliance for Verified Access](#).

To learn whether an AWS service is within the scope of specific compliance programs, see [AWS services in Scope by Compliance Program](#) and choose the compliance program that you are interested in. For general information, see [AWS Compliance Programs](#).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

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

- [Security Compliance & Governance](#) – These solution implementation guides discuss architectural considerations and provide steps for deploying security and compliance features.
- [HIPAA Eligible Services Reference](#) – Lists HIPAA eligible services. Not all AWS services are HIPAA eligible.
- [AWS Compliance Resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [AWS Customer Compliance Guides](#) – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map

the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).

- [Evaluating Resources with Rules](#) in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see [Security Hub controls reference](#).
- [Amazon GuardDuty](#) – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.
- [AWS Audit Manager](#) – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Resilience in Verified Access

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](#).

In addition to the AWS global infrastructure, Verified Access offers the following feature to help support your high availability needs.

Multiple subnets for high availability

When you create a load balancer type Verified Access endpoint, you can associate multiple subnets to the endpoint. Each subnet that you associate with the endpoint must belong to a different Availability Zone. By associating multiple subnets you can ensure high availability by using multiple Availability Zones.

Monitoring AWS Verified Access

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

- **Access logs** – Capture detailed information about requests to access applications. For more information, see [the section called “Verified Access logs”](#).
- **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 section called “CloudTrail logs”](#).

Verified Access logs

After AWS Verified Access evaluates each access request, it logs all access attempts. This provides you with centralized visibility into application access, and helps you quickly respond to security incidents and audit requests. Verified Access supports the Open Cybersecurity Schema Framework (OCSF) logging format.

When you enable logging, you need to configure a destination for the logs to be sent. The IAM principal being used to configure the logging destination needs to have certain permissions for logging to work properly. The required IAM permissions for each logging destination can be seen in the [Verified Access logging permissions](#) section. Verified Access supports the following destinations for publishing access logs:

- Amazon CloudWatch Logs log groups
- Amazon S3 buckets
- Amazon Data Firehose delivery streams

Contents

- [Verified Access logging versions](#)
- [Verified Access logging permissions](#)
- [Enable or disable Verified Access logs](#)

- [Enable or disable Verified Access trust context](#)
- [OCSF version 0.1 log examples for Verified Access](#)
- [OCSF version 1.0.0-rc.2 log examples for Verified Access](#)

Verified Access logging versions

By default, the Verified Access logging system uses Open Cybersecurity Schema Framework (OCSF) version 0.1. For sample logs that use version 0.1 see [OCSF version 0.1 log examples for Verified Access](#).

The latest logging version is compatible with OCSF version 1.0.0-rc.2. For more information about the schema, see [OCSF Schema](#). For sample logs that use version 1.0.0-rc.2, see [OCSF version 1.0.0-rc.2 log examples for Verified Access](#).

Note that you can't use OCSF version 0.1 if the Verified Access endpoint uses the TCP protocol.

To upgrade the logging version using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access instances**.
3. Select the appropriate Verified Access instance.
4. On the **Verified Access instance logging configuration** tab, choose **Modify Verified Access instance logging configuration**.
5. Select **ocsf-1.0.0-rc.2** from the **Update log version** drop-down list.
6. Choose **Modify Verified Access instance logging configuration**.

To upgrade the logging version using the AWS CLI

Use the [modify-verified-access-instance-logging-configuration](#) command.

Verified Access logging permissions

The IAM principal being used to configure the logging destination needs to have certain permissions for logging to work properly. The following sections show the permissions required for each logging destination.

For delivery to CloudWatch Logs:

- `ec2:ModifyVerifiedAccessInstanceLoggingConfiguration` on the Verified Access instance
- `logs:CreateLogDelivery`, `logs>DeleteLogDelivery`, `logs:GetLogDelivery`, `logs:ListLogDeliveries`, and `logs:UpdateLogDelivery` on all resources
- `logs:DescribeLogGroups`, `logs:DescribeResourcePolicies`, and `logs:PutResourcePolicy` on the destination log group

For delivery to Amazon S3:

- `ec2:ModifyVerifiedAccessInstanceLoggingConfiguration` on the Verified Access instance
- `logs:CreateLogDelivery`, `logs>DeleteLogDelivery`, `logs:GetLogDelivery`, `logs:ListLogDeliveries`, and `logs:UpdateLogDelivery` on all resources
- `s3:GetBucketPolicy` and `s3:PutBucketPolicy` on the destination bucket

For delivery to Firehose:

- `ec2:ModifyVerifiedAccessInstanceLoggingConfiguration` on the Verified Access instance
- `firehose:TagDeliveryStream` on all resources
- `iam:CreateServiceLinkedRole` on all resources
- `logs:CreateLogDelivery`, `logs>DeleteLogDelivery`, `logs:GetLogDelivery`, `logs:ListLogDeliveries`, and `logs:UpdateLogDelivery` on all resources

Enable or disable Verified Access logs

You can use the procedures in this section to enable or disable logging. When you enable logging, you need to configure a destination for the logs to be sent. The IAM principal that is used to configure the logging destination needs to have certain permissions for logging to work properly. The required IAM permissions for each logging destination can be seen in the [Verified Access logging permissions](#) section.

Contents

- [Enable access logs](#)
- [Disable access logs](#)

Enable access logs

To enable Verified Access logs

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access instances**.
3. Select the Verified Access instance.
4. On the **Verified Access instance logging configuration** tab, choose **Modify Verified Access instance logging configuration**.
5. (Optional) To include trust data sent from trust providers in the logs, do the following:
 - a. Select **ocsf-1.0.0-rc.2** from the **Update log version** drop-down list.
 - b. Choose **Include trust context**.
6. Do one of the following:
 - Turn on **Deliver to Amazon CloudWatch Logs**. Choose the destination log group.
 - Turn on **Deliver to Amazon S3**. Enter the name, owner, and prefix of the destination bucket.
 - Turn on **Deliver to Firehose**. Choose the destination delivery stream.
7. Choose **Modify Verified Access instance logging configuration**.

To enable Verified Access logs using the AWS CLI

Use the [modify-verified-access-instance-logging-configuration](#) command.

Disable access logs

You can disable access logs for your Verified Access instance at any time. After you disable access logs, your log data remains in your log destination until you delete it.

To disable Verified Access logs

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access instances**.

3. Select the Verified Access instance.
4. On the **Verified Access instance logging configuration** tab, choose **Modify Verified Access instance logging configuration**.
5. Turn off log delivery.
6. Choose **Modify Verified Access instance logging configuration**.

To disable Verified Access logs using the AWS CLI

Use the [modify-verified-access-instance-logging-configuration](#) command.

Enable or disable Verified Access trust context

The trust context sent from your trust provider can optionally be enabled for inclusion in your Verified Access logs. This can be useful when defining policies that allow or deny access to your applications. After you enable it, the trust context is found in the log under the `data` field. If trust context is disabled, the `data` field is set to `null`. To configure Verified Access to include trust context in the logs, do the following procedure.

Note

Including trust context in your Verified Access logs requires upgrading to the latest logging version `ocsf-1.0.0-rc.2`. The following procedure assumes that you already have logging enabled. If that is not true, see [Enable access logs](#) for the full procedure.

Contents

- [Enable trust context](#)
- [Disable trust context](#)

Enable trust context

To include trust context in the Verified Access logs using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access instances**.
3. Select the appropriate Verified Access instance.

4. On the **Verified Access instance logging configuration** tab, choose **Modify Verified Access instance logging configuration**.
5. Select **ocsf-1.0.0-rc.2** from the **Update log version** drop-down list.
6. Turn on **Include trust context**.
7. Choose **Modify Verified Access instance logging configuration**.

To include trust context in the Verified Access logs using the AWS CLI

Use the [modify-verified-access-instance-logging-configuration](#) command.

Disable trust context

If you no longer want to include trust context in the logs, you can remove it by doing the following procedure.

To remove trust context from the Verified Access logs using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Verified Access instances**.
3. Select the appropriate Verified Access instance.
4. On the **Verified Access instance logging configuration** tab, choose **Modify Verified Access instance logging configuration**.
5. Turn off **Include trust context**.
6. Choose **Modify Verified Access instance logging configuration**.

To remove trust context from the Verified Access logs using the AWS CLI

Use the [modify-verified-access-instance-logging-configuration](#) command.

OCSF version 0.1 log examples for Verified Access

The following are sample logs using OCSF version 0.1.

Examples

- [Access granted with OIDC](#)
- [Access granted with OIDC and JAMF](#)
- [Access granted with OIDC and CrowdStrike](#)

- [Access denied due to a missing cookie](#)
- [Access denied by policy](#)
- [Unknown log entry](#)

Access granted with OIDC

In this example log entry, Verified Access allows access to an endpoint with an OIDC user trust provider.

```
{
  "activity": "Access Granted",
  "activity_id": "1",
  "category_name": "Application Activity",
  "category_uid": "8",
  "class_name": "Access Logs",
  "class_uid": "208001",
  "device": {
    "ip": "10.2.7.68",
    "type": "Unknown",
    "type_id": 0
  },
  "duration": "0.004",
  "end_time": "1668580194344",
  "time": "1668580194344",
  "http_request": {
    "http_method": "GET",
    "url": {
      "hostname": "hello.app.example.com",
      "path": "/",
      "port": 443,
      "scheme": "https",
      "text": "https://hello.app.example.com:443/"
    }
  },
  "user_agent": "python-requests/2.28.1",
  "version": "HTTP/1.1"
},
"http_response": {
  "code": 200
},
"identity": {
  "authorizations": [
    {
```

```
        "decision": "Allow",
        "policy": {
            "name": "inline"
        }
    },
    "idp": {
        "name": "user",
        "uid": "vatp-09bc4cbce2EXAMPLE"
    },
    "user": {
        "email_addr": "johndoe@example.com",
        "name": "Test User Display",
        "uid": "johndoe@example.com",
        "uuid": "00u6wj48l1bxTAEXAMPLE"
    }
},
"message": "",
"metadata": {
    "uid": "Root=1-63748362-6408d24241120b942EXAMPLE",
    "logged_time": 1668580281337,
    "version": "0.1",
    "product": {
        "name": "Verified Access",
        "vendor_name": "AWS"
    }
},
"ref_time": "2022-11-16T06:29:54.344948Z",
"proxy": {
    "ip": "192.168.34.167",
    "port": 443,
    "svc_name": "Verified Access",
    "uid": "vai-002fa341aeEXAMPLE"
},
"severity": "Informational",
"severity_id": "1",
"src_endpoint": {
    "ip": "172.24.57.68",
    "port": "48234"
},
"start_time": "1668580194340",
"status_code": "100",
"status_details": "Access Granted",
"status_id": "1",
```

```
"status": "Success",
"type_uid": "20800101",
"type_name": "AccessLogs: Access Granted",
"unmapped": null
}
```

Access granted with OIDC and JAMF

In this example log entry, Verified Access allows access to an endpoint with both OIDC and JAMF device trust providers.

```
{
  "activity": "Access Granted",
  "activity_id": "1",
  "category_name": "Application Activity",
  "category_uid": "8",
  "class_name": "Access Logs",
  "class_uid": "208001",
  "device": {
    "ip": "10.2.7.68",
    "type": "Unknown",
    "type_id": 0,
    "uid": "41b07859-4222-4f41-f3b9-97dc1EXAMPLE"
  },
  "duration": "0.347",
  "end_time": "1668804944086",
  "time": "1668804944086",
  "http_request": {
    "http_method": "GET",
    "url": {
      "hostname": "hello.app.example.com",
      "path": "/",
      "port": 443,
      "scheme": "h2",
      "text": "https://hello.app.example.com:443/"
    },
    "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36",
    "version": "HTTP/2.0"
  },
  "http_response": {
    "code": 304
  },
}
```



```
"identity": {
  "authorizations": [
    {
      "decision": "Allow",
      "policy": {
        "name": "inline"
      }
    }
  ],
  "idp": {
    "name": "oidc",
    "uid": "vatp-9778003bc2EXAMPLE"
  },
  "user": {
    "email_addr": "johndoe@example.com",
    "name": "Test User Display",
    "uid": "johndoe@example.com",
    "uuid": "4f040d0f96becEXAMPLE"
  }
},
"message": "",
"metadata": {
  "uid": "Root=1-321318ce-6100d340adf4fb29dEXAMPLE",
  "logged_time": 1668805278555,
  "version": "0.1",
  "product": {
    "name": "Verified Access",
    "vendor_name": "AWS"
  }
},
"ref_time": "2022-11-18T20:55:44.086480Z",
"proxy": {
  "ip": "10.5.192.96",
  "port": 443,
  "svc_name": "Verified Access",
  "uid": "vai-3598f66575EXAMPLE"
},
"severity": "Informational",
"severity_id": "1",
"src_endpoint": {
  "ip": "192.168.20.246",
  "port": 61769
},
"start_time": "1668804943739",
```

```
"status_code": "100",
"status_details": "Access Granted",
"status_id": "1",
"status": "Success",
"type_uid": "20800101",
"type_name": "AccessLogs: Access Granted",
"unmapped": null
}
```

Access granted with OIDC and CrowdStrike

In this example log entry, Verified Access allows access to an endpoint with both OIDC and CrowdStrike device trust providers.

```
{
  "activity": "Access Granted",
  "activity_id": "1",
  "category_name": "Application Activity",
  "category_uid": "8",
  "class_name": "Access Logs",
  "class_uid": "208001",
  "device": {
    "ip": "10.2.173.3",
    "os": {
      "name": "Windows 11",
      "type": "Windows",
      "type_id": 100
    },
    "type": "Unknown",
    "type_id": 0,
    "uid": "122978434f65093aee5dfbdc0EXAMPLE",
    "hw_info": {
      "serial_number": "751432a1-d504-fd5e-010d-5ed11EXAMPLE"
    }
  },
  "duration": "0.028",
  "end_time": "1668816620842",
  "time": "1668816620842",
  "http_request": {
    "http_method": "GET",
    "url": {
      "hostname": "test.app.example.com",
      "path": "/",

```

```
    "port": 443,
    "scheme": "h2",
    "text": "https://test.app.example.com:443/"
  },
  "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36",
  "version": "HTTP/2.0"
},
"http_response": {
  "code": 304
},
"identity": {
  "authorizations": [
    {
      "decision": "Allow",
      "policy": {
        "name": "inline"
      }
    }
  ],
  "idp": {
    "name": "oidc",
    "uid": "vatp-506d9753f6EXAMPLE"
  },
  "user": {
    "email_addr": "johndoe@example.com",
    "name": "Test User Display",
    "uid": "johndoe@example.com",
    "uuid": "23bb45b16a389EXAMPLE"
  }
},
"message": "",
"metadata": {
  "uid": "Root=1-c16c5a65-b641e4056cc6cb0eeEXAMPLE",
  "logged_time": 1668816977134,
  "version": "0.1",
  "product": {
    "name": "Verified Access",
    "vendor_name": "AWS"
  }
},
"ref_time": "2022-11-19T00:10:20.842295Z",
"proxy": {
  "ip": "192.168.144.62",
```

```
    "port": 443,
    "svc_name": "Verified Access",
    "uid": "vai-2f80f37e64EXAMPLE"
  },
  "severity": "Informational",
  "severity_id": "1",
  "src_endpoint": {
    "ip": "10.14.173.3",
    "port": 55706
  },
  "start_time": "1668816620814",
  "status_code": "100",
  "status_details": "Access Granted",
  "status_id": "1",
  "status": "Success",
  "type_uid": "20800101",
  "type_name": "AccessLogs: Access Granted",
  "unmapped": null
}
```

Access denied due to a missing cookie

In this example log entry, Verified Access denies access due to a missing authentication cookie.

```
{
  "activity": "Access Denied",
  "activity_id": "2",
  "category_name": "Application Activity",
  "category_uid": "8",
  "class_name": "Access Logs",
  "class_uid": "208001",
  "device": null,
  "duration": "0.0",
  "end_time": "1668593568259",
  "time": "1668593568259",
  "http_request": {
    "http_method": "POST",
    "url": {
      "hostname": "hello.app.example.com",
      "path": "/dns-query",
      "port": 443,
      "scheme": "h2",
      "text": "https://hello.app.example.com:443/dns-query"
    }
  }
}
```

```
    },
    "user_agent": "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML",
    "version": "HTTP/2.0"
  },
  "http_response": {
    "code": 302
  },
  "identity": null,
  "message": "",
  "metadata": {
    "uid": "Root=1-5cf1c832-a565309ce20cc7dafEXAMPLE",
    "logged_time": 1668593776720,
    "version": "0.1",
    "product": {
      "name": "Verified Access",
      "vendor_name": "AWS"
    }
  },
  "ref_time": "2022-11-16T10:12:48.259762Z",
  "proxy": {
    "ip": "192.168.34.167",
    "port": 443,
    "svc_name": "Verified Access",
    "uid": "vai-108ed7a672EXAMPLE"
  },
  "severity": "Informational",
  "severity_id": "1",
  "src_endpoint": {
    "ip": "10.7.178.16",
    "port": "46246"
  },
  "start_time": "1668593568258",
  "status_code": "200",
  "status_details": "Authentication Denied",
  "status_id": "2",
  "status": "Failure",
  "type_uid": "20800102",
  "type_name": "AccessLogs: Access Denied",
  "unmapped": null
}
```

Access denied by policy

In this example log entry, Verified Access denies an authenticated request because the request is not allowed by the access policies.

```
{
  "activity": "Access Denied",
  "activity_id": "2",
  "category_name": "Application Activity",
  "category_uid": "8",
  "class_name": "Access Logs",
  "class_uid": "208001",
  "device": {
    "ip": "10.4.133.137",
    "type": "Unknown",
    "type_id": 0
  },
  "duration": "0.023",
  "end_time": "1668573630978",
  "time": "1668573630978",
  "http_request": {
    "http_method": "GET",
    "url": {
      "hostname": "hello.app.example.com",
      "path": "/",
      "port": 443,
      "scheme": "h2",
      "text": "https://hello.app.example.com:443/"
    },
    "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36",
    "version": "HTTP/2.0"
  },
  "http_response": {
    "code": 401
  },
  "identity": {
    "authorizations": [],
    "idp": {
      "name": "user",
      "uid": "vatp-e048b3e0f8EXAMPLE"
    },
    "user": {
```

```

        "email_addr": "johndoe@example.com",
        "name": "Test User Display",
        "uid": "johndoe@example.com",
        "uuid": "0e1281ad3580aEXAMPLE"
    }
},
"message": "",
"metadata": {
    "uid": "Root=1-531a036a-09e95794c7b96aefbEXAMPLE",
    "logged_time": 1668573773753,
    "version": "0.1",
    "product": {
        "name": "Verified Access",
        "vendor_name": "AWS"
    }
},
"ref_time": "2022-11-16T04:40:30.978732Z",
"proxy": {
    "ip": "3.223.34.167",
    "port": 443,
    "svc_name": "Verified Access",
    "uid": "vai-021d5eaed2EXAMPLE"
},
"severity": "Informational",
"severity_id": "1",
"src_endpoint": {
    "ip": "10.4.133.137",
    "port": "31746"
},
"start_time": "1668573630955",
"status_code": "300",
"status_details": "Authorization Denied",
"status_id": "2",
"status": "Failure",
"type_uid": "20800102",
"type_name": "AccessLogs: Access Denied",
"unmapped": null
}

```

Unknown log entry

In this example log entry, Verified Access can't generate a complete log entry so it emits an unknown log entry. This ensures that every request appears in the access log.

```
{
  "activity": "Unknown",
  "activity_id": "0",
  "category_name": "Application Activity",
  "category_uid": "8",
  "class_name": "Access Logs",
  "class_uid": "208001",
  "device": null,
  "duration": "0.004",
  "end_time": "1668580207898",
  "time": "1668580207898",
  "http_request": {
    "http_method": "GET",
    "url": {
      "hostname": "hello.app.example.com",
      "path": "/",
      "port": 443,
      "scheme": "https",
      "text": "https://hello.app.example.com:443/"
    },
    "user_agent": "python-requests/2.28.1",
    "version": "HTTP/1.1"
  },
  "http_response": {
    "code": 200
  },
  "identity": null,
  "message": "",
  "metadata": {
    "uid": "Root=1-435eb955-6b5a1d529343f5adaEXAMPLE",
    "logged_time": 1668580579147,
    "version": "0.1",
    "product": {
      "name": "Verified Access",
      "vendor_name": "AWS"
    }
  },
  "ref_time": "2022-11-16T06:30:07.898344Z",
  "proxy": {
    "ip": "10.1.34.167",
    "port": 443,
    "svc_name": "Verified Access",
    "uid": "vai-6c32b53b3cEXAMPLE"
  }
}
```



```
    },
    "severity": "Informational",
    "severity_id": "1",
    "src_endpoint": {
      "ip": "172.28.57.68",
      "port": "47220"
    },
    },
    "start_time": "1668580207893",
    "status_code": "000",
    "status_details": "Unknown",
    "status_id": "0",
    "status": "Unknown",
    "type_uid": "20800100",
    "type_name": "AccessLogs: Unknown",
    "unmapped": null
  }
}
```

OCSF version 1.0.0-rc.2 log examples for Verified Access

The following are sample logs using OCSF version 1.0.0-rc.2.

Examples

- [Access granted with trust context included](#)
- [Access granted with trust context omitted](#)
- [Assign privileges with network CIDR endpoint](#)

Access granted with trust context included

```
{
  "activity_name": "Access Grant",
  "activity_id": "1",
  "actor": {
    "authorizations": [{
      "decision": "Allow",
      "policy": {
        "name": "inline"
      }
    }],
    "idp": {
      "name": "user",
      "uid": "vatp-09bc4cbce2EXAMPLE"
    }
  }
}
```

```
    },
    "invoked_by": "",
    "process": {},
    "user": {
      "email_addr": "johndoe@example.com",
      "name": "Test User Display",
      "uid": "johndoe@example.com",
      "uuid": "00u6wj48l bxTAEXAMPLE"
    },
    "session": {}
  },
  "category_name": "Audit Activity",
  "category_uid": "3",
  "class_name": "Access Activity",
  "class_uid": "3006",
  "device": {
    "ip": "10.2.7.68",
    "type": "Unknown",
    "type_id": 0
  },
  "duration": "0.004",
  "end_time": "1668580194344",
  "time": "1668580194344",
  "http_request": {
    "http_method": "GET",
    "url": {
      "hostname": "hello.app.example.com",
      "path": "/",
      "port": 443,
      "scheme": "https",
      "text": "https://hello.app.example.com:443/"
    }
  },
  "user_agent": "python-requests/2.28.1",
  "version": "HTTP/1.1"
},
"http_response": {
  "code": 200
},
"message": "",
"metadata": {
  "uid": "Root=1-63748362-6408d24241120b942EXAMPLE",
  "logged_time": 1668580281337,
  "version": "1.0.0-rc.2",
  "product": {
```

```
        "name": "Verified Access",
        "vendor_name": "AWS"
    }
},
"ref_time": "2022-11-16T06:29:54.344948Z",
"proxy": {
    "ip": "192.168.34.167",
    "port": 443,
    "svc_name": "Verified Access",
    "uid": "vai-002fa341aeEXAMPLE"
},
"severity": "Informational",
"severity_id": "1",
"src_endpoint": {
    "ip": "172.24.57.68",
    "port": "48234"
},
"start_time": "1668580194340",
"status_code": "100",
"status_detail": "Access Granted",
"status_id": "1",
"status": "Success",
"type_uid": "300601",
"type_name": "Access Activity: Access Grant",
"data": {
    "context": {
        "oidc": {
            "family_name": "Last",
            "zoneinfo": "America/Los_Angeles",
            "exp": 1670631145,
            "middle_name": "Middle",
            "given_name": "First",
            "email_verified": true,
            "name": "Test User Display",
            "updated_at": 1666305953,
            "preferred_username": "johndoe-user@test.com",
            "profile": "http://www.example.com",
            "locale": "US",
            "nickname": "Tester",
            "email": "johndoe-user@test.com",
            "additional_user_context": {
                "aud": "xxx",
                "exp": 10000000000,
                "groups": [
```

```

        "group-id-1",
        "group-id-2"
    ],
    "iat": 1000000000,
    "iss": "https://oidc-tp.com/",
    "sub": "xyzsubject",
    "ver": "1.0"
  }
},
"http_request": {
  "x_forwarded_for": "1.1.1.1,2.2.2.2",
  "http_method": "GET",
  "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36",
  "port": "80",
  "hostname": "hostname.net"
}
}
}
}

```

Access granted with trust context omitted

```

{
  "activity_name": "Access Grant",
  "activity_id": "1",
  "actor": {
    "authorizations": [{
      "decision": "Allow",
      "policy": {
        "name": "inline"
      }
    }],
    "idp": {
      "name": "user",
      "uid": "vatp-09bc4cbce2EXAMPLE"
    },
    "invoked_by": "",
    "process": {},
    "user": {
      "email_addr": "johndoe@example.com",
      "name": "Test User Display",
      "uid": "johndoe@example.com",

```

```
      "uuid": "00u6wj48l1bxTAEXAMPLE"
    },
    "session": {}
  },
  "category_name": "Audit Activity",
  "category_uid": "3",
  "class_name": "Access Activity",
  "class_uid": "3006",
  "device": {
    "ip": "10.2.7.68",
    "type": "Unknown",
    "type_id": 0
  },
  "duration": "0.004",
  "end_time": "1668580194344",
  "time": "1668580194344",
  "http_request": {
    "http_method": "GET",
    "url": {
      "hostname": "hello.app.example.com",
      "path": "/",
      "port": 443,
      "scheme": "https",
      "text": "https://hello.app.example.com:443/"
    }
  },
  "user_agent": "python-requests/2.28.1",
  "version": "HTTP/1.1"
},
"http_response": {
  "code": 200
},
"message": "",
"metadata": {
  "uid": "Root=1-63748362-6408d24241120b942EXAMPLE",
  "logged_time": 1668580281337,
  "version": "1.0.0-rc.2",
  "product": {
    "name": "Verified Access",
    "vendor_name": "AWS"
  }
},
"ref_time": "2022-11-16T06:29:54.344948Z",
"proxy": {
  "ip": "192.168.34.167",
```

```

    "port": 443,
    "svc_name": "Verified Access",
    "uid": "vai-002fa341aeEXAMPLE"
  },
  "severity": "Informational",
  "severity_id": "1",
  "src_endpoint": {
    "ip": "172.24.57.68",
    "port": "48234"
  },
  "start_time": "1668580194340",
  "status_code": "100",
  "status_detail": "Access Granted",
  "status_id": "1",
  "status": "Success",
  "type_uid": "300601",
  "type_name": "Access Activity: Access Grant",
  "data": null
}

```

Assign privileges with network CIDR endpoint

```

{
  "activity_id": "1",
  "activity_name": "Assign Privileges",
  "category_name": "Audit Activity",
  "category_uid": "3",
  "class_name": "Authorization",
  "class_uid": "3003",
  "data": {
    "endpoint_type": "cidr",
    "protocol": "tcp",
    "access_path": "public",
    "idp": {
      "name": "my-oidc-instance",
      "uid": "vatp-09bc4cbce2EXAMPLE"
    },
    "authorizations": [{
      "decision": "Allow",
      "policy": {
        "name": "inline"
      }
    }],
  }
}

```

```
"context": {
  "oidc": {
    "family_name": "Last",
    "zoneinfo": "America/Los_Angeles",
    "exp": 1670631145,
    "middle_name": "Middle",
    "given_name": "First",
    "email_verified": true,
    "name": "Test User Display",
    "updated_at": 1666305953,
    "preferred_username": "johndoe-user@test.com",
    "profile": "http://www.example.com",
    "locale": "US",
    "nickname": "Tester",
    "email": "johndoe-user@test.com",
    "additional_user_context": {
      "aud": "xxx",
      "exp": 1000000000,
      "groups": [
        "group-id-1",
        "group-id-2"
      ],
      "iat": 1000000000,
      "iss": "https://oidc-tp.com/",
      "sub": "xyzsubject",
      "ver": "1.0"
    }
  },
  "tcp_flow": {
    "destination_ip": "10.0.0.1",
    "destination_port": 22,
    "client_ip": "10.2.7.68"
  }
},
"device": {
  "ip": "10.2.7.68",
  "port": 1002,
  "type": "Unknown",
  "type_id": 0
},
"duration": "0.004",
"end_time": "1668580194344",
"time": "1668580194344",
```

```
{
  "metadata": {
    "uid": "",
    "logged_time": 1668580281337,
    "version": "1.0.0-rc.2",
    "product": {
      "name": "Verified Access",
      "vendor_name": "AWS"
    }
  },
  "severity": "Informational",
  "severity_id": "1",
  "start_time": "1668580194340",
  "status_code": "200",
  "status_id": "1",
  "status": "Success",
  "type_uid": "300301",
  "type_name": "Authorization: Assign Privileges",
  "count": 1,
  "dst_endpoint": {
    "ip": "107.22.231.155",
    "port": 22
  },
  "privileges": [
    "vae-12345cbce2EXAMPLE"
  ],
  "user": {
    "email_addr": "johndoe-user@test.com",
    "uid": "johndoe-user",
    "uuid": "9bcce02a-fc15-4091-a0b7-874d157c67b8"
  }
}
```

Log Verified Access API calls using AWS CloudTrail

AWS Verified Access is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Verified Access. CloudTrail captures API calls for Verified Access as events. The calls captured include calls from the Verified Access console and code calls to the Verified Access API operations. Using the information collected by CloudTrail, you can determine the request that was made to Verified Access, the IP address from which the request was made, when it was made, and additional details.

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

- Whether the request was made with root user or user credentials.
- Whether the request was made on behalf of an IAM Identity Center user.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

CloudTrail is active in your AWS account when you create the account and you automatically have access to the CloudTrail **Event history**. The CloudTrail **Event history** provides a viewable, searchable, downloadable, and immutable record of the past 90 days of recorded management events in an AWS Region. For more information, see [Working with CloudTrail Event history](#) in the *AWS CloudTrail User Guide*. There are no CloudTrail charges for viewing the **Event history**.

For an ongoing record of events in your AWS account past 90 days, create a trail or a [CloudTrail Lake](#) event data store.

CloudTrail trails

A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. All trails created using the AWS Management Console are multi-Region. You can create a single-Region or a multi-Region trail by using the AWS CLI. Creating a multi-Region trail is recommended because you capture activity in all AWS Regions in your account. If you create a single-Region trail, you can view only the events logged in the trail's AWS Region. For more information about trails, see [Creating a trail for your AWS account](#) and [Creating a trail for an organization](#) in the *AWS CloudTrail User Guide*.

You can deliver one copy of your ongoing management events to your Amazon S3 bucket at no charge from CloudTrail by creating a trail, however, there are Amazon S3 storage charges. For more information about CloudTrail pricing, see [AWS CloudTrail Pricing](#). For information about Amazon S3 pricing, see [Amazon S3 Pricing](#).

CloudTrail Lake event data stores

CloudTrail Lake lets you run SQL-based queries on your events. CloudTrail Lake converts existing events in row-based JSON format to [Apache ORC](#) format. ORC is a columnar storage format that is optimized for fast retrieval of data. Events are aggregated into *event data stores*, which are immutable collections of events based on criteria that you select by applying [advanced event selectors](#). The selectors that you apply to an event data store control which events persist

and are available for you to query. For more information about CloudTrail Lake, see [Working with AWS CloudTrail Lake](#) in the *AWS CloudTrail User Guide*.

CloudTrail Lake event data stores and queries incur costs. When you create an event data store, you choose the [pricing option](#) you want to use for the event data store. The pricing option determines the cost for ingesting and storing events, and the default and maximum retention period for the event data store. For more information about CloudTrail pricing, see [AWS CloudTrail Pricing](#).

Verified Access management events

[Management events](#) provide information about management operations that are performed on resources in your AWS account. These are also known as control plane operations. By default, CloudTrail logs management events.

Verified Access logs control plane operations as management events. For a list, see the [Amazon EC2 API Reference](#).

Verified Access event examples

The following example shows a CloudTrail event that demonstrates the `CreateVerifiedAccessInstance` action.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAIKK400INJWEXAMPLE:jdope",
    "arn": "arn:aws:iam::123456789012:user/jdope",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "jdope"
  },
  "eventTime": "2022-11-18T20:44:04Z",
  "eventSource": "ec2.amazonaws.com",
  "eventName": "CreateVerifiedAccessInstance",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "198.51.100.1",
  "userAgent": "console.amazonaws.com",
  "requestParameters": {
    "CreateVerifiedAccessInstanceRequest": {
```

```
        "Description": "",
        "ClientToken": "85893b1e-49f6-4d24-97de-280c664edf1b"
    },
    "responseElements": {
        "CreateVerifiedAccessInstanceResponse": {
            "verifiedAccessInstance": {
                "creationTime": "2022-11-18T20:44:04",
                "description": "",
                "verifiedAccessInstanceId": "vai-0d79d91875542c549",
                "verifiedAccessTrustProviderSet": ""
            },
            "requestId": "2eae195d-6bfd-46d7-b46e-a68cb791de09"
        }
    },
    "requestID": "2eae195d-6bfd-46d7-b46e-a68cb791de09",
    "eventID": "297d6529-1144-40f6-abf8-3a76f18d88f0",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

For information about CloudTrail record contents, see [CloudTrail record contents](#) in the *AWS CloudTrail User Guide*.

Quotas for AWS Verified Access

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific.

AWS account-level quotas

Your AWS account has the following quotas related to Verified Access.

Name	Default	Adjustable	Description
Verified Access Instances	5	Yes	The maximum number of Verified Access Instances that customers can create in the current Region.
Verified Access Groups	10	Yes	The maximum number of Verified Access Groups that customers can create in the current Region.
Verified Access Trust Providers	15	Yes	The maximum number of Verified Access Trust Providers that customers can create in the current Region.
Verified Access Endpoints	50	Yes	The maximum number of Verified Access Endpoints that customers can create in the current Region.

HTTP headers

The following are the size limits for HTTP headers.

Name	Default	Adjustable
Request line	16 K	No
Single header	16 K	No

Name	Default	Adjustable
Entire response header	32 K	No
Entire request header	64 K	No

HTTP traffic

The connection idle timeout is 60 seconds. If an application takes longer than 60 seconds to respond to an HTTP request, the client receives an HTTP 504 gateway timeout error. If Verified Access logs is enabled, we log any HTTP 504 errors.

OIDC claim size

The following is the OIDC claim size limit.

Name	Default	Adjustable
OIDC claim size	11 K	No

IAM Identity Center

Verified Access can provide access to users in IAM Identity Center who are assigned to up to 1,000 groups.

Document history for the Verified Access User Guide

The following table describes the documentation releases for Verified Access.

Change	Description	Date
Support for access tokens in the trust context	Update to add additional <code>l_user_context</code> to OIDC user claims.	February 24, 2025
Support for resources over non-HTTP protocols	Release of access to resources over non-HTTP protocols.	February 5, 2025
Preview release	Preview release of access to resources over non-HTTP protocols.	December 1, 2024
AWS managed policy updated	Update made to AWS managed IAM policy for Verified Access.	November 17, 2023
Data encryption at rest	AWS Verified Access encrypts data at rest by default, using AWS owned KMS keys.	September 28, 2023
Support for FIPS compliance	Configure Verified Access for FIPS compliance.	September 26, 2023
Enhanced logging	Addition of logging feature which adds trust contexts to logs.	June 19, 2023
AWS managed policy updated	Update made to AWS managed IAM policy for Verified Access.	May 31, 2023

[GA release](#)

GA release of the Verified Access User Guide. Includes [AWS WAF integration](#).

April 27, 2023

[Preview release](#)

Preview release of the Verified Access User Guide

November 29, 2022