PingAccess®

Version 3.0

Overview

Ping Identity

Copyright

PingAccess

Version 3.0
July, 2014

Ping Identity Corporation
1001 17th Street, Suite 100
Denver, CO 80202
U.S.A.

Trademark

Disclaimer

Document Lifetime

Ping Identity may occasionally update online documentation between releases of the related software. Consequently, if this PDF was not downloaded recently, it may not contain the most up-to-date information. Please refer to the online documentation at *documentation.pingidentity.com* for the most current information.

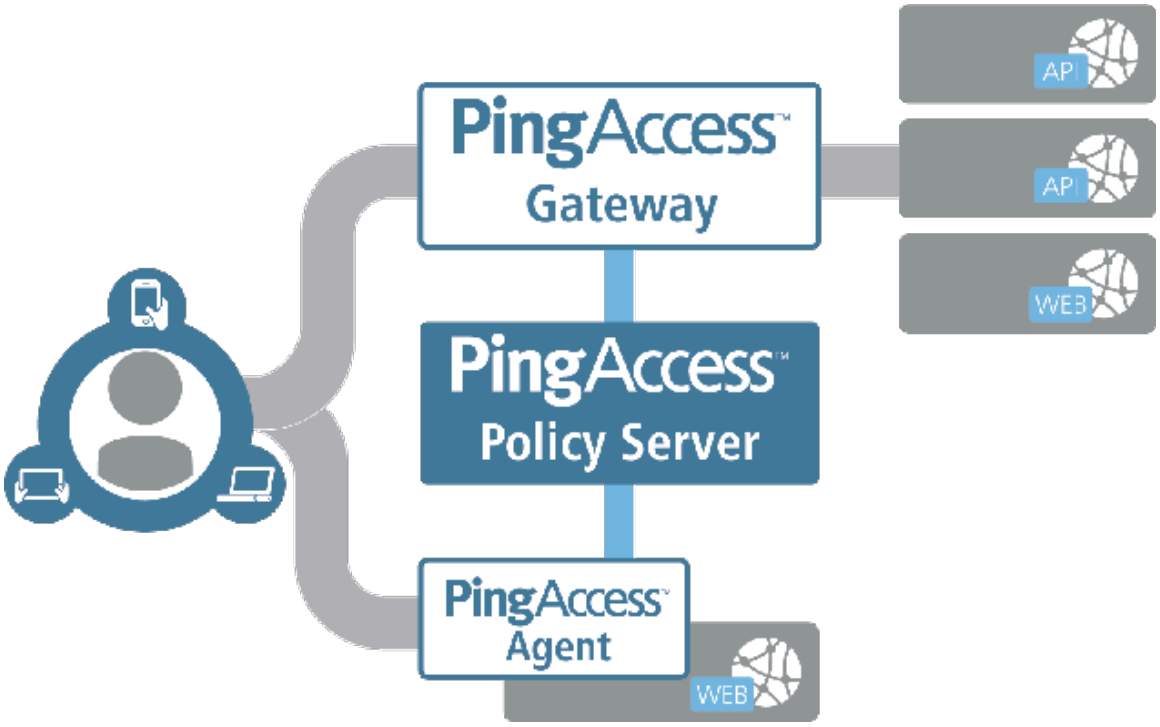
From the web site, you may also download and refresh this PDF if it has been updated, as indicated by a change on this date: **July, 2014**.

Overview

PingAccess Overview

PingAccess protects Web Applications and APIs by applying security policies to client requests to determine if access is allowed. Requests can either be routed through PingAccess Gateway to the target Site or be intercepted at the target Web server by a PingAccess Agent which in turn communicates with PingAccess Policy Server. In either case, policies specified for the target Application are evaluated and PingAccess grants or denies access. When access is granted, the client request can be modified to provide additional identity information as needed by the target Application.

- Sites represent applications and APIs running on Web servers to be protected by PingAcess. They are defined by host and port settings to which PingAccess will forward authorized client requests.
- Agents represent Web server plugins deployed on Web servers to be protected by PingAccess. Agents contact PingAccess for authorization before allowing client requests to access the target assets.
- Applications represent applications and APIs which clients need to access securely and specify the information needed to protect them. Applications are composed of resources which have distinct access control requirements. Access rules can be applied to applications and their resources for greater flexibility.
- Policies are rules applied to applications and resources and determine if access is allowed. Rules are evaluated in the context of the client's identity and request characteristics.



Web Access Management

Web Access Management

With growing numbers of internal and external users, and more and more enterprise resources available online, it is important to ensure that qualified users can access only those resources to which they have permission. PingAccess uses Web Access Management (WAM) capabilities to allow organizations to manage access rights to Web-based resources.

WAM is a form of identity management that controls access to Web resources, providing authentication and policy-based access management. Once a user is authenticated, PingAccess applies application and resource-level policies to the request. Once policy evaluation is passed, any required identity mediation between the back-end site and the authenticated user is performed. The user is then granted access to the requested resource.

PingAccess 3.0 provides two deployment architectures for Web Access Management - gateway and agent. In a gateway deployment client requests are routed to PingAccess which then forwards authorized requests to the target application. In an agent deployment, client requests are

intercepted at the web server hosting the application via the PingAccess agent plugin. The agent then communicates with PingAccess Policy Server to validate access before allowing the request to proceed to the target application resource.

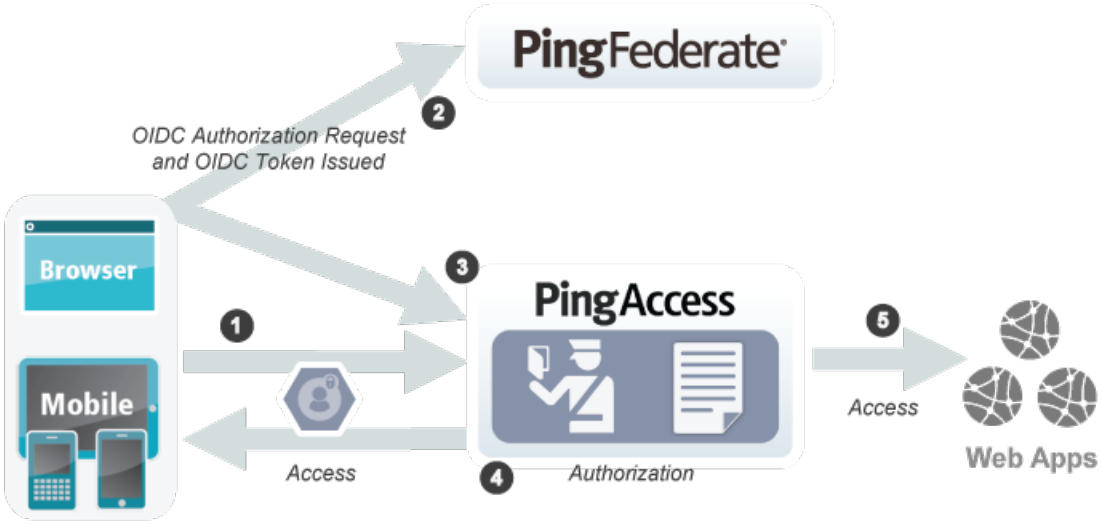
Related Topics

Web Session Initiation
Application Scoped Web Sessions

Web Session Initiation

WAM Session Initiation

Once a user authenticates, PingAccess applies the application and resource-level policies to the request. Once policy evaluation is passed, any required token mediation between the back-end Site and the authenticated user is performed. The user is then granted access to the Site



Processing Steps:

1. When a user requests a Web resource from PingAccess, PingAccess inspects the request for a PA Token.
2. If the PA Token is missing, PingAccess redirects the user to an OpenID Connect Provider (OP) for authentication.

> When using an OP, an OAuth Client must already be configured in PingAccess. For steps on configuring an OAuth Client within PingFederate, see Configuring a Client. To then configure that OAuth Client within PingAccess, see the Web Session section on the PingFederate page.

3. The OP follows the appropriate authentication process, evaluates domain-level policies, and issues an OpenID Connect (OIDC) ID Token to PingAccess.
4. PingAccess validates the ID Token and issues a PA Token and sends it to the browser in a cookie during a redirect to the original target resource. Upon gaining access to the resource, PingAccess evaluates application and resource-level policies and optionally audits the request.

> PingAccess can perform Token Mediation by exchanging the PA Token for the appropriate security token from the PingFederate STS or from a cache (if token mediation occurred recently).

5. PingAccess forwards the request to the target site.
6. PingAccess processes the response from the site to the browser (step not shown).

> See the Web Sessions section for more information.

Application Scoped Web Sessions

Application Scoped Web Sessions

PingAccess Tokens can be configured to have their Web Sessions scoped to a specific application. This improves the security model of the session by preventing unrelated applications from impersonating the end user.

Several controls exist to scope the PA Token to an application:

- **Audience Attribute**: The audience attribute defines who the token is applicable to and is represented as a short, unique identifier. Requests are rejected that contain a PA Token with an audience that differs from what is configured in the Web Session associated with the target Resource.
- **Audience Suffix**: The audience attribute is also used as a suffix of the cookie name to ensure uniqueness--for example, PA.businessAppAudience.
- **Cookie Domain**: The cookie domain can also optionally be set to limit where the PA Token is sent.

> In addition to these controls, parameters such as session timeout can be adjusted to match the policy requirements of each application.

Corresponding OAuth clients must be defined in PingFederate for each Web Session. Redirect URL whitelists defined in PingFederate dictate from which servers and domains the session can originate. Controlling this within PingFederate enables flexibility of the attribute contract (and its fulfillment) for that particular application. This ensures that each application and its associated policies only deal with attributes related to it.

Server-Side Session Management

Server-Side Session Management

The server-side session management feature allows for tighter session control, leveraging the single logout capabilities provided by PingFederate 7.2. The ability to enforce single logout enables the following scenarios:

1. PingAccess can reject a PingAccess cookie associated with a session that has been typically based on end user driven logout.
2. The end user can initiate a logout from all PingAccess issued web sessions using a centralized logout.

This feature performs a validation check with PingFederate when protected resources are served. The OpenID Connect option must be enabled in the OAuth 2.0 Authorization Server (AS) role, and access to the OpenID Connect session revocation API must be enabled.

Token Mediation

Token Mediation

The differing identity needs of applications across an organization present a number of challenges. User identity and credential systems can vary widely, making the task of managing and mapping identity difficult. PingAccess provides token mediation to address these identity requirements in a gateway deployment. For agent deployments, similar functionality is available via Identity Mappings.

PingAccess performs token mediation by acquiring the appropriate security token from the PingFederate STS or from a cache (if identity mediation occurred recently). The STS provides security-token validation and creation, exchanging the PA Token or an OAuth Bearer Token for the appropriate security token--for example, Oracle Access Manager, OpenToken, or custom tokens. The token exchanged depends on which Site Authenticator is configured for the Site within PingAccess. Site Authenticators define the type of authentication the Site requires.

The following Site Authenticators integrate with a variety of security models, including:

- HTTP Basic Authentication
- Mutual TLS
- OpenToken

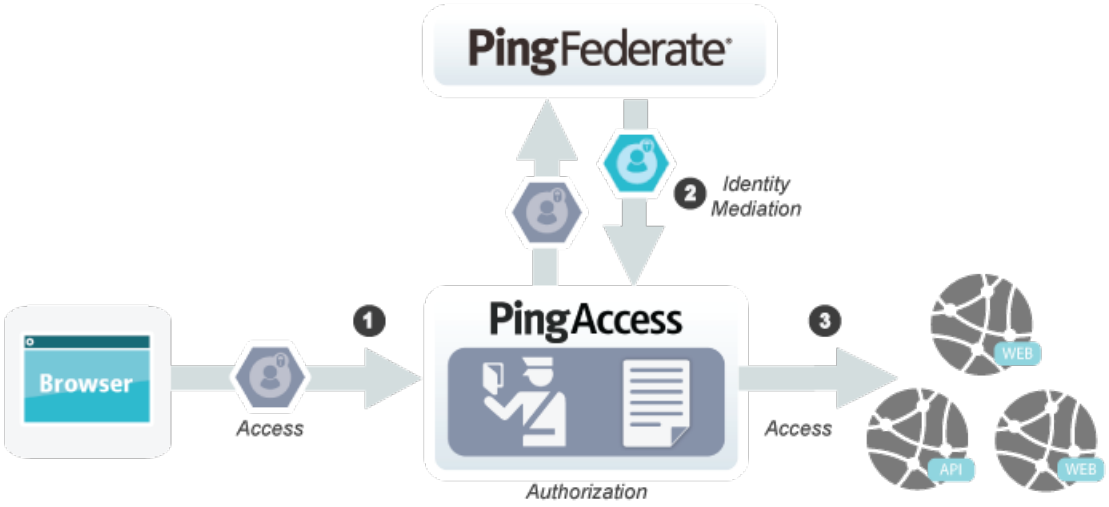
Related Topics

Token Mediation Flow

Token Mediation Flow

Token Mediation Flow

The following illustration shows an example of token mediation using PingFederate to exchange a PA Token or OAuth Bearer Token for a different security token.



Processing Steps:

1. A user requests a Resource from PingAccess with a PA Token or OAuth Bearer Token.

   > This example assumes the user has already obtained a PA Token or OAuth Bearer Token. See Web Access Management or Using the OAuth Authorization Server for details on how users authenticate with PingFederate and obtain a PA Token or OAuth Bearer Token.

2. PingAccess evaluates resource-level policies and performs token mediation by acquiring the appropriate security token from the PingFederate STS (shown in the diagram) or from a cache (if token mediation occurred recently) specified by the Site Authenticator.
3. PingAccess sends the request to the Site (Web application) with the appropriate token.
4. PingAccess returns the response to the client (not shown).

Using Virtual Hosts

Using Virtual Hosts

Virtual Hosting enables you to host multiple server or domain names. This allows one server to share resources without requiring all sites on the server to use the same host name--for example, you may want to use multiple names on the same server so that each site name reflects the services offered rather than the actual server name where those sites are hosted.

PingAccess supports virtual hosting by serving requests bound for a set of defined server names and mapping them to requested applications. The target host header presented by the client can optionally be rewritten with the appropriate back-end host name. For example, say the host configured for Site One is `hr121.internal:80`. You configure Application One to use a virtual host of `hr.mycompany.com:80`. You associate Site One with Application One. PingAccess listens for incoming requests for the site at `hr.mycompany.com:80`. When a client request comes in to `hr.mycompany.com:80`, PingAccess sees the request, looks at the name of the domain configured for the back-end site, and replaces the target Host header with `hr121.internal:80`.

Supporting HTTPS requests causes additional complexity due to the need for SSL/TLS certificates. Prior to availability of SNI in Java 8, an HTTPS port could only present a single certificate. In order to handle multiple Virtual Hosts you have to use a wildcard name certificate or the Subject Alternative Name (SAN) extension. With SNI available, Virtual Hosts can present different certificates on a single HTTPS port. You can assign which certificates (Key Pairs) are used by which Virtual Host on the HTTPS Listeners page - see HTTPS Listeners.

Clustering

Clustering

PingAccess provides clustering features that allow a group of PingAccess servers to appear as a single system. When deployed appropriately, server clustering can facilitate high availability of critical services. Clustering can also increase performance and overall system throughput. It is important to understand, however, that availability and performance are often at opposite ends of the deployment spectrum. Thus, you may need to make some configuration tradeoffs that balance availability with performance to accommodate specific deployment goals.

In a cluster, you can configure each PingAccess engine, or node, as either an administrative console or a runtime engine in the run.properties file.

Runtime engines service client requests, while the console server administers policy and configuration for the entire cluster (via the administrative console). A cluster may contain one or more runtime nodes, but only one console node. Server-specific configuration data is stored in the PingAccess administrative console server in the run.properties file. Information needed to bootstrap an engine is stored in the bootstrap.properties file on each engine.

At startup, a PingAccess engine node in a cluster checks its local configuration and then makes a call to the administrative console to check for changes. How often each engine in a cluster checks the console for changes is configurable in the engine run.properties file.

Configuration information is replicated to all engine nodes. By default, engines do not share runtime state. For increased performance, you can configure engines to share runtime state by configuring cluster interprocess communication using the run.properties file (see Cluster Configuration Settings).

> Runtime state clustering consists solely of a shared cache of security tokens acquired from the PingFederate STS for Token Mediation use cases using the Token Mediator Site Authenticator.

Related Topics

Configure PingAccess Servers Into a Cluster

Using the OAuth Authorization Server

Using the OAuth Authorization Server

PingAccess supports the Bearer Token Security Model and the Validation Grant Type extension grant and uses an OAuth AS in the following ways:

- Works with OAuth Authorization Servers such as the PingFederate OAuth AS to authorize access to protected Resources.
- Protects applications by requiring an OAuth bearer access token (see Section 2.1 of RFC 6750 for supported token transport details).
- Acts as an OAuth Resource Server, requesting validation from the OAuth AS for the bearer access token it receives from a client making a protected-resources call. The OAuth AS validates the access token and sends token attributes to PingAccess, which evaluates the returned OAuth details against policies set by mapping Rules to Resources.
- Grants access to a Resource based on the use of Rules in combination with the OAuth AS validation.