PingDirectory™

Release 7.0

Consent Solution Guide

Notice

PingDirectory™ Product Documentation

Trademarks

Disclaimer

Support

https://support.pingidentity.com/

Contents

Chapter 1

Introduction to the Consent Service and Consent API

Topics:

Companies gain loyalty and trust when they offer transparency and control to their users and customers regarding the personal data that is collected, processed, or shared. In Europe, the General Data Protection Regulation (GDPR) was designed specifically for allowing companies to collect and use valuable data about its users, while protecting the rights of citizens to control what is collected and used. To support the collection and end-user control of personal data, PingDirectory Server includes schema and REST APIs that provide the ability to collect fine-grained data authorizations (consents), from users and customers.

Consent Service overview

The Consent Service is an HTTP-based REST API hosted by the PingDirectory Server or PingDirectoryProxy Server. The service enables the collection of consent from application users, the enforcement of consent, a user's management of his or her consent, and auditing of consent actions. Enterprises can integrate these features into their applications to give users transparency and control of their data privacy.

For the purpose of this document, the following terms are used:

Table 1: Consent Terms

Term	Meaning
Consent definition	The terms of the fine-grained contract, which describes the data that can be processed or shared, and a purpose for processing or sharing the data. The consent definition is stored in the server configuration.
Consent localization	A child object of a definition that contains versioned, localized text for the consent definition, to be used when prompting an individual. This is stored in the server configuration.
Consent record	A record of a consent interaction with a user. Consent records are stored in the directory tree.
Subject	The individual whose data can be collected, processed, or shared.
Actor	The individual who granted/denied/revoked consent. This is usually the same as the subject.
Audience	The entity, application, or service that is granted or denied access to a subject's data for a specific purpose.

Consent API overview

The PingDirectory Server and PingDirectoryProxy Server provide a REST API for managing individuals' consent to handle their data. This can be used as a component of a larger solution, such as a GDPR compliance system or the PingDataGovernance Server Open Banking Account Requests API.

The PingDirectory Server Consent API enables authorization services:

- to capture user consent for sharing or processing data
- to confirm that consent to share or process data has been granted
- for individuals to manage the consent that they have granted.

Detailed API documentation can be found on the Ping Identity website.

How consents are collected

User consent is collected by creating a consent record through the Consent API. In most cases, the Consent API client uses consent localization data to construct an approval prompt to display to the user. This prompt should include text describing what data is collected and for what purpose, allowing the user to make an informed decision about the value of sharing his or her data.

For example, a web application needing to collect consent for a user's browsing behavior would use the Consent API to look up the localizations for the `browsing-behavior` consent definition. It would select the localization appropriate for the user and use data from the localization resource to construct a consent prompt for display to the user. After the user is prompted and makes a decision, the client could store the decision by creating a new consent record through the Consent API.

How consents are enforced

The Consent Service can be used as a data source for making access control decisions. If a particular data usage scenario requires consent, then the application or service needing to access or process that data must not be able to use the data unless the user has provided consent. The entity that performs this consent check may be the application itself or some other service.

To perform a consent check, the Consent API client must be able to correlate a data access request type with a consent definition. For example, if a web application needs to collect a user's browsing behavior, this data collection scenario might be represented by a consent definition called `browsing-behavior`. The application would check for an existing consent grant by searching the Consent API for a consent record that matches the user and the `browsing-behavior` consent definition. If a match is found, then the application can proceed. If a match is not found, the application must collect consent from the user.

How applications use the Consent API

The following example illustrates both consent capture and consent enforcement. This example follows a user's journey on a website during which the company must gather consent to track the user's browsing behavior:

1. A user launches the company's application and authenticates. The application wants to record the page visit, but first it must check if the user has granted consent to do so.
2. The application makes a call to the Consent API to determine if the `browsing-behavior` consent record exists for this user, and whether consent been granted.
3. The API returns a result indicating that no consent record exists. The application must prompt the user for his or her consent. The application calls the Consent API to retrieve the localization for the `browsing-behavior` consent, which includes the language that the application uses to produce a prompt for the user.
4. After the user makes a decision, the application stores the user's decision by creating a new consent record. This is through a call to the Consent API.
5. Later, the user visits another page in the company's site. The application wants to record the page visit, and again checks whether the user has granted consent to do so.
6. The application makes a call to the Consent API to get the `browsing-behavior` consent record for this user.
7. If the user's consent record agrees to have the company track his or her browsing behavior, the application can then make the appropriate calls to track browsing behavior. This is consent enforcement.

Chapter 2

Consent Service configuration

Topics:

This section provides details for installing and configuring the components on which the Consent Service relies. Refer to the PingDirectory Server Administration Guide for detailed configuration information.

Configuration overview

The Consent Service is not enabled by default. The setup and configuration process varies depending on the following factors:

- Whether client applications will allow an individual to self-manage consents.
- Whether some or all client applications will be privileged, with the ability to manage all consents.
- The HTTP authentication method used by client applications.
- Whether consent records exist in the same directory as user entries.

Example configuration scenarios

The following client application scenarios are available for determining how the Consent Service should be configured to meet your business needs.

Directly managed consents

In this scenario, one or more client applications provide provide an interface for individuals to directly manage their own consent records. These applications can only manage consents for the currently authenticated user. In addition, there is also a client application for consent administrators. An OAuth 2 authorization server grants access tokens that the applications uses to access the Consent API.

Configuration for this scenario includes:

1. Configure an OAuth 2 authorization server to issue a `urn:pingdirectory:consent` scope to individuals and a `urn:pingdirectory:consent_admin` scope to consent administrators.
2. Create an identity mapper to map subject identifiers used by the authorization server to LDAP DNs used by the PingDirectory Server.
3. Configre an access token validator to validate tokens issued by the OAuth 2 authorization server.
4. Configure the Consent HTTP Servlet Extension to disable HTTP basic authentication and restart the HTTPS Connection Handler.
5. Configure the Consent Service to use the OAuth scopes, token validator, and identity mapper.

Indirectly managed consents (basic authentication)

In this scenario, an application uses a privileged service account to manage its users' consents. The application's privileged account can access any consent record, which gives the application the ability to perform operations that an individual user cannot. The following include steps the setup needed for the PingDataGovernance Server's Open Banking Account Requests service to use the Consent Service as its backend.

Configuration for this scenario includes:

1. Create a service account for the application.
2. Configure the Consent HTTP Servlet Extension to enable HTTP basic authentication and restart the HTTPS Connection Handler.
3. Create an identity mapper to map consent record subject and actor attribute values to LDAP DNs. This is optional.
4. Configure the Consent Service to use the application's service account, and optionally the identity mapper.

Set up with the configuration scripts

PingDirectory Server includes two configuration scripts that can serve as the starting point for setting up the Consent Service. Both scripts must be carefully reviewed and updated to support your client application scenarios and business needs.

- `consent-service-base-entries.ldif` - This LDIF script can be imported to create the base DN where consent records will be stored.
- `consent-service-cfg.dsconfig` - This script can be imported to configure and enable the Consent Service.

Both are located in the `/resource/consent/` directory of the PingDirectory Server server root.

Basic configuration with the `consent-service-base-entries.ldif` file includes:

1. Edit the LDIF script and change the location of where consent records will be stored.
2. Import the LDIF script using the `ldapmodify`command, such as:

```
$ bin/ldapmodify --defaultAdd \
  --filename consent-service-base-entries.ldif
```

Basic configuration with the `consent-service-cfg.dsconfig` file includes:

1. Search for `CHANGE-ME` and replace values.
2. Review configuration commands and make additional changes to match existing Ping environment parameters, application scenarios, and business needs.
3. Impost the script with the `dsconfig` command, such as:

```
$ bin/dsconfig --no-prompt \
  --batch-file consent-service-cfg.dsconfig
```

Setup in a replicated PingDirectory Server environment

Running the Consent Service setup script requires special consideration in an environment that includes replicated PingDirectory Servers. If possible, setup the Consent Service after replication is enabled for the PingDirectory Servers. See the *PingDirectory Server Administration Guide* for details about server replication.

Set up Consent Service after replication is enabled

Complete the following steps if replication is already enabled for PingDirectory Servers.

1. If needed, configure the PingDirectory Servers to use a configuration group called "all-servers." This will ensure that configuration changes are applied to all servers in a topology.

```
$ bin/dsconfig set-global-configuration-prop \
  --set configuration-server-group:all-servers
```

2. Run the Consent Service setup script.

```
$ bin/dsconfig --no-prompt \
  --batch-file resource/consent/consent-service-cfg.dsconfig
```

Set up Consent Service before replication is enabled

If you have already set up the Consent Service on a standalone PingDirectory Server, perform the following the steps before enabling replication. In this example, "DS1" is the original PingDirectory Server, and "DS2" is the second server that will be added as a replica.

1. Run the `config-diff` command without arguments on DS1 to produce a batch file that contains configuration changes that will be applied to DS2.

```
$ bin/config-diff > config-changes.dsconfig
```

2. Apply the `config-changes.dsconfig` file to DS2.

```
$ bin/dsconfig --no-prompt \
  --batch-file config-changes.dsconfig \
  --applyChangeTo single-server
```

3. Restart DS2.
4. Enable replication between the two servers.

Configuration reference

There are many configuration options for the Consent Sevice and application integration. The configuration scripts included with the PingDirectory Server provide a starting point. Additional detailed information about the Consent Service properties and configuration is provided as reference.

General Consent Service configuration

The Consent Service configuration is used to control authorization behavior and determines where consent records are stored in the PingDirectory Server. The service properties are configured with the `dsconfig set-consent-service-prop` command. The consent service configuration script configures the consent service properties as follows:

```
$ bin/dsconfig set-consent-service-prop \
  --set enabled:true \
  --set base-dn:ou=consents,dc=example,dc=com \
  --set "bind-dn:cn=consent service account" \
  --set unprivileged-consent-scope:urn:pingdirectory:consent \
  --set privileged-consent-scope:urn:pingdirectory:consent_admin \
  --set "consent-record-identity-mapper:User ID Identity Mapper"
```

The following are Consent Service properties.

Table 2: Consent Service properties

Property	Description	Required to enable service
`enabled`	If set to true, enables the Consent Service for handling client requests.	Yes
`base-dn`	Specifies a container DN for consent record entries.	Yes
`bind-dn`	Specifies an internal service account used by the Consent Service to perform LDAP operations.	Yes
`service-account-dn`	Specifies one or more DNs of requesters that will be considered privileged when using basic authentication. If not defined, a requester will only be considered privileged if it is mapped to a DN with the `bypass-acl` privilege. Optional.	No
`unprivileged-consent-scope`	Specifies the name of the scope required for bearer tokens representing unprivileged requesters.	Yes
`privileged-consent-scope`	Specifies the name of the scope required for bearer tokens representing privileged requesters.	Yes
`consent-record-identity-mapper`	Specifies one or more identity mappers used to map consent record `subject` and `actor` values to DNs. If not defined,	No

Property	Description	Required to enable service
	the Consent Service will not handle requests from unprivileged clients. Optional.	
`audience`	Specifies an `audience` claim value that the Consent Service will require to be present in bearer tokens that it accepts. Optional.	No

For the Consent Service to report itself as unavailable to clients, the following must be true:

- The Consent Service must be enabled.
- The Consent Service base DN must be configured and must exist.
- The internal service account must be configured and must exist.
- The internal service account must have the right to read, add, modify, and delete entries under the Consent Service base DN.

Create a container entry for consent records

Each consent record is a distinct entry in the PingDirectory Server, and the Consent Service requires that these entries be stored under a common base DN, defined by the `base-dn` property of the Consent Service configuration. The Consent Service LDIF file sets the base DN. Use these steps to choose a different location to store consent records.

1. To create the Consent Service base DN, open a text editor and save the following to the file `consent-service-base-dn.ldif`.

```
dn: ou=consents,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
ou: consents
```

2. Use `ldapmodify` to add the entry.

```
$ bin/ldapmodify --defaultAdd --filename consent-service-base-dn.ldif
```

Create an internal service account

The Consent Service uses an internal LDAP connection to operate against consent records that are stored as LDAP entries. It authenticates this LDAP connection using a service account, which must be created and dedicated solely to the Consent Service.

The Consent Service configuration script configures the internal service account using a topology admin user. If needed, this can be changed to a root DN user or a user DN whose entry is in the user backend. In all cases, the service account should exist in every LDAP server in the topology.

This service account must have full read and write access to the Consent Service base DN, the ability to read users' `isMemberOf` attribute, and the right to use the following LDAP controls:

- IntermediateClientRequestControl (1.3.6.1.4.1.30221.2.5.2)
- NameWithEntryUUIDRequestControl (1.3.6.1.4.1.30221.2.5.44)
- RejectUnindexedSearchRequestControl (1.3.6.1.4.1.30221.2.5.54)
- PermissiveModifyRequestControl (1.2.840.113556.1.4.1413)
- PostReadRequestControl (1.3.6.1.1.13.2)

For more information about configuring access, see the *"Managing Access Contro"l* chapter of the PingDirectory Server Administration Guide.

- To ensure the correct access, create a user with the `bypass-acl` privilege. The following `dsconfig` command creates a topology admin user with the `bypass-acl` privilege. After this is created, set this user as the `bind-dn` for the Consent Service.

```
$ dsconfig create-topology-admin-user \
  --user-name "Consent Service Account" \
  --set "description:Consent API service account" \
  --set "alternate-bind-dn:cn=consent service account" \
  --set first-name:Consent \
  --set inherit-default-root-privileges:false \
  --set last-name:Service \
  --set password:CHANGE-ME \
  --set privilege:bypass-acl
```

- Because the `bypass-acl` privilege grants a broad level of access, you may not want to grant this privilege to the Consent Service account. If desired, add the following ACI to enable a targetted set of functionality for the Consent Service. The following example grants this access to the DN `cn=consent service account` using global ACIs:

```
# Grant access to the consent record base DN ou=consents,dc=example,dc=com
dsconfig set-access-control-handler-prop --add 'global-aci:(target="ldap:///
ou=consents,dc=example,dc=com")(targetattr="*||+")(version 3.0; acl "Consent
 Service account access to consent record data"; allow(all) userdn="ldap:///
cn=consent service account";)'

# Grant access to the LDAP request controls used by the Consent Service.
dsconfig set-access-control-handler-prop --add 'global-aci:
(targetcontrol="1.3.6.1.4.1.30221.2.5.2||1.3.6.1.4.1.30221.2.5.44||
1.3.6.1.4.1.30221.2.5.54||1.2.840.113556.1.4.1413||1.3.6.1.1.13.2")(version
 3.0; acl "Consent Service account access to selected controls"; allow
 (read) userdn="ldap:///cn=consent service account";)'

# Grant access to use the Effective Rights Control.
dsconfig set-access-control-handler-prop --add 'global-aci:
(targetcontrol="1.3.6.1.4.1.42.2.27.9.5.2")(version 3.0; acl "Consent
 Service account access to Get Effective Rights control"; allow (read)
 userdn="ldap:///cn=consent service account";)'
```

Configure an identity mapper

The Consent Service uses identity mappers to map requester identities, subject values, and actor values to DNs. An identity mapper takes a user identifier string and correlates the identifier with the DN of a user entry. The PingDirectory Server provides four different types of identity mappers.

Table 3: Identity mappers

Identity mapper type	Description
Exact match identity mapper	Maps a user identifier to a DN by searching for an entry with an attribute that exactly matches the identifier.
Regular expression identity mapper	Similar to an exact match identity mapper, but allows a regular expression to be specified for more flexible matching.
Third-party identity mapper	A custom Java identity mapper implementation written using the Server SDK.
Groovy scripted identity mapper	A custom Groovy identity mapper implementation written using the Server SDK.

The Consent Service can be configured to use identity mappers for each of the following scenarios:

- Requesters authenticating using basic authentication - use the Consent HTTP Servlet Extension `identity-mapper` property to configure an identity mapper that takes the HTTP Basic authorization username string to find the corresponding user's identity in the PingDirectory Server.

- Requesters authenticating using bearer token authentication - use the Access Token Validator `identity-mapper` property to configure an identity mapper that takes the subject (or other claim value from the OAuth token) to find the corresponding user's identity in the PingDirectory Server.
- Consent record actor and subject values - use the Consent Service `consent-record-identity-mapper` property to configure an identity mapper that takes these consent record attribute values and uses them to find the corresponding users' identities in the PingDirectory Server.

The Consent Service configuration script configures a single identity mapper to be used for all three scenarios. The provided identity mapper searches by `uid`, `cn`, or `entryUUID` attributes under the base DNs `cn=config and ou=people,dc=example,dc=com`.

The following configuration provides an example of an identity mapper that will match a user identifier to an LDAP entry with the same value in its `uid` attribute:

```
$ bin/dsconfig create-identity-mapper --mapper-name "User ID Exact Match" \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:uid
```

The following configuration shows another typical example, that of an identity mapper that will match a user identifier to an LDAP entry with the same value in its `entryUUID` attribute:

```
$ bin/dsconfig create-identity-mapper --mapper-name "EntryUUID Exact Match"
 \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:entryUUID
```

The last example creates an identity mapper that will match a user identifier to an LDAP entry with the same value in either its `uid`, `cn`, or `entryUUID` attribute. This identity mapper will also constrain its search to the `ou=people,dc=example,dc=com` and `cn=config` base DNs. (The `cn=config` base DN is not searched by default, and must be explicitly listed to be searched.)

```
$ bin/dsconfig create-identity-mapper \
  --mapper-name "User ID Identity Mapper" \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:uid \
  --set match-attribute:cn \
  --set match-attribute:entryUUID \
  --set match-base-dn:cn=config \
  --set match-base-dn:ou=people,dc=example,dc=com
```

Authentication methods

The Consent Service supports two HTTP authentication methods, which are both enabled by default:

- Basic authentication
- Bearer token authentication

The Consent servlet looks at the request's Authorization header to determine which authentication type is being used by the client.

With basic authentication, the client provides an encoded username/password pair in the HTTP Authorization request header. When the Consent Service receives a request using basic authentication, it maps the username credential to a DN using an identity mapper. This DN is designated the `auth DN` and is used to make subsequent authorization decisions. The Consent Service then performs an LDAP bind using the DN and password to determine if the request can be processed.

With bearer token authentication, the client provides an access token in the HTTP Authorization request header. The access token is always obtained by the client from an external OAuth 2 authorization server and encapsulates information ("claims") about a user identity, the client identity, and the requests that the client is authorized to make.

The PingDirectory Server must be configured to accept access tokens using one or both available access token validators:

- **PingFederate access token validator**. Supports access tokens issued by a PingFederate authorization server. This validator verifies an access token and discovers its claims by making a request to the PingFederate server's token introspection endpoint.
- **JWT access token validator**. Supports signed or encrypted JWT access tokens issued by an arbitrary authorization server. This validator checks an access token by cryptographically verifying the token's signature using a trusted public certificate. The token's claims are encoded in the token itself, so discovering the token's claims does not require an outgoing token introspection request.

The token validator uses its identity mapper to map the subject claim to a DN. This DN is designated the `auth DN` and is used along with the token's claims to make subsequent authorization decisions.

If the PingDirectory Server is configured with at least one access token validator, it will be used by the Consent Service. If the PingDirectory Server is configured with more than one access token validator, the validators are consulted in order until one is able to successfully authenticate the request.

If the PingDirectory Server is configured with multiple access token validators, but only one should be used by the Consent Service, the access token validator can be configured by setting the `access-token-validator` property of the Consent HTTP Servlet Extension.

Note: Configuring an access token validator for the Consent Service requires information from the authorization server configuration:

- The values that the authorization server sets for `subject` claims must be mappable to a DN in the PingDirectory Server.
- The authorization server must be configured to authorize clients and grant scopes appropriately for privileged or unprivileged Consent API access.
- The authorization server must be configured to issue tokens with scopes corresponding to the Consent Service's `unprivileged-scope-name` and `privileged-scope-name` configuration.

Refer to the authorization server's documentation for guidance.

Configure basic authentication

Basic authentication is enabled by default, and the settings are configured in the Consent HTTP Servlet Extension configuration.

- Use the following command to disable basic authentication.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set basic-auth-enabled:false
```

- Use the following command to enable basic authentication.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set basic-auth-enabled:true
```

- Use the following command to configure an identity mapper for basic authentication.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set "identity-mapper:User ID Exact Match"
```

- All of these configuration changes require the Consent servlet to be reloaded before they can take effect. Use the following commands to restart the connection handler that hosts the Consent servlet.

```
$ bin/dsconfig set-connection-handler-prop \
```

```
  --handler-name "HTTPS Connection Handler" \
  --set enabled:false
```

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --set enabled:true
```

Configure bearer token authentication

- The following is an example access token validator configured on the PingDirectory Server for a PingFederate server:

```
$ bin/dsconfig create-external-server \
  --server-name PingFederate \
  --type http \
  --set base-url:https://my-ping-federate-server:1443/
```

```
$ bin/dsconfig create-access-token-validator \
  --validator-name "PingFederate Token Validator" \
  --type ping-federate \
  --set enabled:true \
  --set "identity-mapper:User ID Exact Match" \
  --set authorization-server:PingFederate \
  --set client-id:id \
  --set client-secret:secret
```

- If more than one access token validator is configured on the PingDirectory Server, the Consent Service can be configured to use a single validator with the following command:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set "access-token-validator:PingFederate Token Validator"
```

Configure Consent Service scopes

The Consent Service checks access tokens for a `subject` claim and uses an identity mapper to map the value to a DN, called the request DN or auth DN. If no request DN can be mapped, the request is rejected. In addition, the Consent Service will only accept an access token with a scope that it is configured to recognize.

- An unprivileged consent scope designates the requester as unprivileged. The scope's name is configured with the Consent Service's `unprivileged-consent-scope` property.
- A privileged consent scope designates the requester as privileged. This is configured using the Consent Service's `privileged-consent-scope` property.

The authorization server must also be configured to issue tokens with these scopes.

- The following example configures these scopes for the Consent Service.

```
$ bin/dsconfig set-consent-service-prop \
  --set privileged-users-group-dn:dn_name \
  --set unprivileged-consent-scope:consent \
  --set privileged-consent-scope:consent_admin
```

Authorization

The Consent Service's distinction between privileged and unprivileged requesters determines the type of operations that can be performed by requesters. During the authorization phase, the Consent servlet performs checks on both the bearer token claims (if present) and the `auth DN` to determine if the requester is privileged or unprivileged. These are summarized in the following table.

Table 4: Available operations per requester type

Requester type	Description	Access determined by	Can create consent records	Can update consent records	Can delete consent records
Unprivileged	Requesters with no authority to operate on consent records other than their own. Unprivileged requesters can only be used if the Consent Service is configured to map subject and actor DNs using the `consent-record-identity-mapper` property.	A requester is considered unprivileged if it does not meet any of the criteria for a privileged requester. If using bearer token authentication, the access token must include a scope named by the `unprivileged-consent-scope` property of the Consent Service configuration. Also, an unprivileged requester can only perform actions on consent records where the subject DN and actor DN match the requester DN.	Yes, if the requester DN matches the subject DN and actor DN.	Yes, if the requester DN matches the subject DN and actor DN.	No.
Privileged	A requester with the authority to perform any operation on any consent record.	When using basic authentication, a requester is considered privileged if the requester DN either has the `bypass-acl` privilege or is listed in the `service-account-dn` property of the Consent Service configuration. If using bearer token authentication, the access token must include a scope named by the `privileged-consent-scope` property of the Consent Service configuration.	Yes.	Yes.	Yes.

Bearer token check

If a bearer token was used, the following checks are performed:

- If the Consent Service's `audience` property is configured, the bearer token's audience claim must match the configured value.
- If the bearer token contains a scope matching the Consent Service's `privileged-scope-name` property, then the requester is considered privileged.
- If not, the bearer token must have a scope matching the Consent Service's `unprivileged-scope-name` property, and the requester is considered unprivileged.

Basic authentication check

If basic authentication is used, the following checks are performed:

- If the `auth DN` has the LDAP privilege `bypass-acl`, the requester is privileged.
- If the `auth DN` is listed in the Consent Service's `service-account-dn` property, the requester is privileged.
- If not, the requester is considered unprivileged.

Chapter 3

Manage consents

Topics:

This section describes the tasks required to support the collection and end-user control of personal data, and manage users' consents.

Overview of consent management

The full lifecycle of consent management goes beyond collecting the user's consent. First, the terms of each consent contract must be centrally managed. After collecting consent, the user will want to review previously granted consents and potentially revoke some. Finally, companies will need to be able to trace the history of updates to any consent in order to resolve a dispute or respond to audit.

Consent definitions and localizations

Companies will want to centrally manage the language used when prompting a user to give consent. This is key to ensuring a consistent user experience across multiple applications, such as mobile and web. The Consent Service requires one or more consent definitions to be defined in the PingDirectory Server configuration. Each consent definition represents the combination of:

- The data to be collected or shared.
- The purpose for collecting or sharing this data.

For example, a consent definition could represent user email addresses, used to deliver a third party's email newsletter. A consent definition could also represent access to a user's network-connected IoT device, which would be used for a home automation task controlled by a third party.

Each consent definition must have one or more localization. A localization is a versioned object consisting of the data that a Consent API client needs to prompt a user for consent. When a consent record is accepted or denied by a Consent Service client, it must include a reference to a consent definition, locale, and version.

Create consent definition and localization

- The following creates a consent definition and a localization for it.

```
$ bin/dsconfig create-consent-definition \
  --definition-name email_newsletter \
  --set "display-name:Email newsletter"
```

```
$ bin/dsconfig create-consent-definition-localization \
  --definition-name email_newsletter \
  --localization-name en-US \
  --set version:1.0 \
  --set "data-text:Your email address" \
  --set "purpose-text:To receive newsletter updates"
```

- The following example updates a localization and its version.

```
$ bin/dsconfig set-consent-definition-localization-prop \
  --definition-name email_newsletter \
  --localization-name en-US \
  --set version:1.1 \
  --set "data-text:Your preferred email address"
```

Perform an audit on consents

Changes to Consent Service resources are tracked by one of two types of audit logs. For examples of configuring either type of log, see the `<server-root>/resource/consent-service-cfg.dsconfig` script bundled with the server or *Logging*. This example uses the Consent Trace Logger. It represents Consent Service change events using the same field names used by the Consent API.

Table 5: Log Publishers

Log publisher	Log publisher type	Description
Consent Trace Logger	file-based-trace	Records Consent Service events at the Consent API level. Change events are recorded using messages of type `audit`.
Consent LDAP Audit Logger	file-based-audit	Records data changes at the LDAP level. In combination with a Request Criteria configuration object, an LDAP audit logger can be configured to record changes to Consent Service resources only.

Trace logger keys for auditing

Trace logger audit messages consist of a timestamp, the message type (`CONSENT AUDIT`), and a set of key/value pairs. A subset of important keys are described in the following table.

Note: The keys used in trace log audit messages vary depending on the type of resource.

Table 6: Log Publishers

Trace logger key	Description
`requestID`	A server-specific HTTP request ID. This value can be correlated with messages produced by other loggers.
`resourceType`	The type of Consent Service resource that was changed. Possible values are `definition`, `localization`, or `consent`.
`changeType`	The type of change recorded by this message. Possible values are `create`, `update`, or `delete`.
`attrsAdded`	A comma-delimited list of the attributes that were added to the resource.
`attrsUpdated`	A comma-delimited list of the attributes that were modified on the resource.
`attrsDeleted`	A comma-delimited list of the attributes that were removed from the resource.
`requestDN`	The DN of the requester, which is available only when the resource type is `consent`.
`definitionID`	The consent definition ID. If the resource type is `definition`, this identifies the definition that was changed. If the resource type is `localization`, this identifies the parent definition. If the resource type is `consent`, this identifies the consent record's related definition.
`locale`	The locale. If the resource type is `localization`, this identifies the localization (in combination with the definition ID). If the resource type is `consent`, this identifies the related localization (combined with the definition ID).
`consentID`	The consent record ID, available only when the resource type is `consent`.
`subject`	The subject value, available only when the resource type is `consent`.
`subjectDN`	The subject's mapped LDAP DN, available only when the resource type is `consent`.
`actor`	The actor value, available only when the resource type is `consent`.

Trace logger key	Description
`actorDN`	The actor's mapped LDAP DN, available only when the resource type is `consent`.
`audience`	The audience value, available only when the resource type is `consent`.
`status`	The consent status. Possible values are `pending`, `accepted`, `denied`, `revoked`, and `restricted`. Only available when the resource type is `consent`.
`previousStatus`	The previous consent status, if applicable. Only available when the resource type is `consent`.
`msg`	A multiline value that includes the complete body of the changed resource. If the action is an update or a delete, the resource's body before the change will be included.

Perform an audit

Consent resource changes for particular entities (such as a specific user, or a specific consent definition) can be audited by searching the trace log using a combination of one of the message keys and the desired value. For example, if an individual's LDAP DN is known, then the `subjectDN` key can be used to construct a text search for any audit log messages containing that DN. Any matching log messages would constitute a history of that individual's consent activity.

Example new consent record

The following is a sample record. this audit log message provides important values in a parseable key/value format, but also includes the entirety of the new consent record.

```
[22/May/2018:18:02:42.584 -0500] CONSENT AUDIT requestID=57
 requestDN="uid=user.0,ou=people,
  dc=example,dc=com" consentID="6cff325b-e092-4094-b7f9-5a30864b0d24"
 subject="user.0" subjectDN="uid=user.0,
  ou=People,dc=example,dc=com" actor="user.0"
 actorDN="uid=user.0,ou=People,dc=example,dc=com" audience="client1"
  definitionID="cats" locale="en-US" status="accepted"
 attrsAdded="actor,audience,createdDate,dataText,subject,
  purposeText,definition,id,updatedDate,actorDN,status,subjectDN"
 changeType="create" resourceType="consent" msg="
New Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'accepted','subject':'user.0','subjectDN':'uid=user.0,

 ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=People,dc=exampl
  'client1','definition':{'id':'cats','version':'1.0','locale':'en-
US'},'dataText':'Collect data about your
  cats','purposeText':'To recommend cat food flavors that will satisfy and
 delight your feline companion',

 'createdDate':'2018-05-22T23:02:42.553Z','updatedDate':'2018-05-22T23:02:42.553Z'}"
```

Example updated consent record

This example shows the complete consent record before and after it was updated. With the `attrsUpdated`, `status`, and `previousStatus` keys, one can determine that the `status` changed from accepted to revoked.

```
[22/May/2018:18:05:08.660 -0500] CONSENT AUDIT requestID=59
 requestDN="uid=user.0,ou=people,
```

```
   dc=example,dc=com" consentID="6cff325b-e092-4094-b7f9-5a30864b0d24"
 subject="user.0" subjectDN="uid=user.0,
   ou=People,dc=example,dc=com" actor="user.0"
 actorDN="uid=user.0,ou=People,dc=example,dc=com"
   audience="client1" definitionID="cats" locale="en-US" status="revoked"
 previousStatus="accepted"
   attrsUpdated="status" changeType="update" resourceType="consent" msg="
Previous Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'accepted','subject':'user.0','subjectDN':'uid=user.0,

 ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=People,dc=exampl
  'audience':'client1','definition':
{'id':'cats','version':'1.0','locale':'en-US'},'dataText':'Collect
  data about your cats','purposeText':'To recommend cat food flavors that
 will satisfy and delight your
  feline
 companion','createdDate':'2018-05-22T23:02:42.553Z','updatedDate':'2018-05-22T23:02:42
Updated Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'revoked','subject':'user.0','subjectDN':

 'uid=user.0,ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=Peop
  dc=com','audience':'client1','definition':
{'id':'cats','version':'1.0','locale':'en-US'},'dataText':
  'Collect data about your cats','purposeText':'To recommend cat food
 flavors that will satisfy and
  delight your feline
 companion','createdDate':'2018-05-22T23:02:42.553Z','updatedDate':'2018-05-22T23:05:08
```

Example deleted consent record

This example shows that a consent record has been deleted, and the complete representation of the consent record prior to its deletion is provided.

```
[22/May/2018:18:06:35.071 -0500] CONSENT AUDIT requestID=61
 requestDN="cn=directory manager"
   consentID="6cff325b-e092-4094-b7f9-5a30864b0d24" subject="user.0"
 subjectDN="uid=user.0,ou=People,
   dc=example,dc=com" actor="user.0"
 actorDN="uid=user.0,ou=People,dc=example,dc=com" audience="client1"
   definitionID="cats" locale="en-US" status="revoked"
 previousStatus="revoked" attrsDeleted="actor,audience,

 createdDate,dataText,subject,purposeText,definition,id,updatedDate,actorDN,status,subje
 changeType="delete"
   resourceType="consent" msg="
Deleted Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'revoked','subject':'user.0','subjectDN':

 'uid=user.0,ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=Peop
   dc=example,dc=com','audience':'client1','definition':
{'id':'cats','version':'1.0','currentVersion':
   '1.0','locale':'en-US'},'dataText':'Collect data about your
 cats','purposeText':'To recommend cat food
   flavors that will satisfy and delight your feline
 companion','createdDate':'2018-05-22T23:02:42.553Z',
   'updatedDate':'2018-05-22T23:05:08.655Z'}"
```

Logging

The PingDirectory Server trace log publisher is used for logging events generated by HTTP service operations. The trace logger can be used to observe, debug, and audit consent requests.

👉 **Note:** To create a log of consent audit events only, remove all message types except for `consent-message-type:audit`.

- The following example of creates a trace logger for all consent events, plus summaries of HTTP requests and responses.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "Consent Trace Logger" \
  --type file-based-trace \
  --set "description:Records Consent API operations" \
  --set enabled:true \
  --set consent-message-type:audit \
  --set consent-message-type:consent-created \
  --set consent-message-type:consent-deleted \
  --set consent-message-type:consent-retrieved \
  --set consent-message-type:consent-search \
  --set consent-message-type:consent-updated \
  --set consent-message-type:definition-created \
  --set consent-message-type:definition-deleted \
  --set consent-message-type:definition-retrieved \
  --set consent-message-type:definition-search \
  --set consent-message-type:definition-updated \
  --set consent-message-type:error \
  --set consent-message-type:localization-created \
  --set consent-message-type:localization-deleted \
  --set consent-message-type:localization-retrieved \
  --set consent-message-type:localization-search \
  --set consent-message-type:localization-updated \
  --set http-message-type:request \
  --set http-message-type:response \
  --set 'exclude-path-pattern:/**/*.css' \
  --set 'exclude-path-pattern:/**/*.eot' \
  --set 'exclude-path-pattern:/**/*.gif' \
  --set 'exclude-path-pattern:/**/*.ico' \
  --set 'exclude-path-pattern:/**/*.jpg' \
  --set 'exclude-path-pattern:/**/*.js' \
  --set 'exclude-path-pattern:/**/*.png' \
  --set 'exclude-path-pattern:/**/*.svg' \
  --set 'exclude-path-pattern:/**/*.ttf' \
  --set 'exclude-path-pattern:/**/*.woff' \
  --set 'exclude-path-pattern:/**/*.woff2' \
  --set 'exclude-path-pattern:/console/**' \
  --set 'exclude-path-pattern:/console/**/template/**' \
  --set log-file:logs/consent-trace \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy"
```

Correlating user and consent data

In some cases, the organization that has been granted consent by a group of users may need to perform an LDAP search so that they can act upon consent data in the aggregate. For example, a marketing group has collected consent to send a newsletter by email. A search must be performed that will list all of the consent records where the consent

definition is `email` and the status is `accepted`. Those records must be correlated to user entries, and each user's email address must be retrieved.

This task is performed with an LDAP search on the PingDirectory Server. Every consent record has a `subject`, the user whose data is collected and stored. The Consent Service can be configured so that it stores the subject's DN in the `subjectDN` field.

In the LDAP schema:

- A consent record's `subjectDN` field is the `ping-consent-subject-dn` attribute.
- A consent record's status is the `ping-consent-state` attribute.
- A consent record's definition ID is in the `ping-consent-definition.id` JSON attribute field.
- And a user entry's email address is in the `mail` attribute.

The search will need to find all of the consent record entries where `ping-consent-definition.id` is `email` and the `ping-consent-status` is `accepted`. It then needs to correlate those consent record entries to user entries using `ping-consent-subject-dn`, and retrieve each user entry's `mail` attribute value. For example:

```
$ bin/ldapsearch \
   --baseDN "ou=consents,dc=example,dc=com" \
   --searchScope sub \
   --joinRule "dn:ping-consent-subject-dn" \
   --joinBaseDN "ou=people,dc=example,dc=com" \
   --joinScope sub \
   --joinRequestedAttribute mail
   '&(ping-consent-
definition:jsonObjectFilterExtensibleMatch:={ "filterType" : "equals",
 "field" : "id", "value" : "email" })(ping-consent-state=accepted)' \
   1.1

   # Join Result Control:
   #      OID:  1.3.6.1.4.1.30221.2.5.9
   #      Join Result Code:  0 (success)
   #      Joined With Entry:
   #           dn: uid=user.0,ou=People,dc=example,dc=com
   #           mail: user.0@example.com
   dn: entryUUID=9e481010-8330-425a-
bbf1-6637de053d48,ou=Consents,dc=example,dc=com

   # Result Code:  0 (success)
   # Number of Entries Returned:  1
```

The output listed under "Join Result Control" specifies the mail value.

Troubleshooting

The following are general guidelines for troubleshooting the Consent Service and any connection issues. When evaluating the configuration, make sure these issues are addressed first:

- Is the Consent Service enabled?
- Does the Consent Service base DN exist?
- Does the Consent Service's service account have the correct permissions?
- If the Consent Service should accept bearer tokens:
  - Are one or more Access Token Validators correctly configured?
  - Are the identity mappers for the Access Token Validators configured correctly?
  - Are the authorization servers correctly configured to issue tokens that the Consent Service will accept? Check the `audience`, `privileged-consent-scope`, and `unprivileged-consent-scope` properties of the Consent Service configuration.

- If privileged users are defined, are the members of the LDAP group specified by the Consent Service configuration's `privileged-users-group-dn` property?
- If there are applications that allow individuals to manage their own consents, is the system properly configured to map `actor` and `subject` DNs? Check the Consent Service configuration's `consent-record-identity-mapper` property.

Error cases

Consent Service is unavailable

If the Consent Service is unavailable, check that the service is enabled and that the communcation with the service is available. Confirm that the service account for the Consent Service has been properly provisioned. If the Consent Service resides on a PingDirectoryProxy Server, make sure that the service account exists on the PingDirectoryProxy Server and all PingDirectory Server behind the PingDirectoryProxy Server.

Requester lacks sufficient rights to perform operation

A request may be rejected with a 403 for the following reasons:

- The bearer token does not contain a required scope. Check the `privileged-consent-scope` and `unprivileged-consent-scope` properties of the Consent Service configuration.
- The bearer token does not contain a required `audience` claim. Check the `audience` property of the Consent Service configuration.
- Authentication was successful, but the requester is unprivileged and attempted to perform an operation that only a privileged requester may perform. For example, it may have attempted to act upon a consent record that it does not own, or it may have attempted to delete a consent record.

When using basic authentication, the requester must be listed in the Consent Service configuration `service-account-dn` property to be considered privileged.

Subject and actor do not match

Only a privileged requester can create or modify a consent record whose `subject` and `actor` values do not match.

Unindexed search

The Consent Service will not allow a client to make an unindexed search. In most cases, a client should be able to fix this by refining the search. For example, if a search by `subject` would be unindexed, perform a search by `subject definition` ID.

Search size limit exceeded

The Consent Service caps the maximum number of records that can be returned in a search result using its `search-size-limit` configuration property. This limit can be increased, or the client may be able to refine the search to produce fewer results.

Index

U