

Cloudera Runtime 7.3.1

Apache Knox Authentication

Date published: 2020-07-28

Date modified: 2024-12-10

CLOUDERA

<https://docs.cloudera.com/>

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

| | |
|---|-----------|
| Apache Knox overview..... | 5 |
| Dynamically generating Knox topology files..... | 5 |
| Securing access to Hadoop cluster: Apache Knox..... | 5 |
| Apache Knox Gateway overview..... | 6 |
| Knox Supported Services Matrix..... | 6 |
| Knox Topology Management in Cloudera Manager..... | 8 |
| Considerations for Knox..... | 9 |
| | |
| Proxy Cloudera Manager through Apache Knox..... | 9 |
| | |
| Installing Apache Knox..... | 10 |
| Apache Knox Install Role Parameters..... | 12 |
| | |
| Management of Knox shared providers in Cloudera Manager..... | 13 |
| Configure Apache Knox authentication for PAM..... | 14 |
| Configure Apache Knox authentication for AD/LDAP..... | 15 |
| Knox CLI testing tools..... | 17 |
| Configure Apache Knox Authentication for SAML..... | 18 |
| Add a new shared provider configuration..... | 18 |
| TLS Mutual Authentication..... | 19 |
| Management of existing Apache Knox shared providers..... | 20 |
| Add a new provider in an existing provider configuration..... | 21 |
| Modify a provider in an existing provider configuration..... | 23 |
| Disable a provider in an existing provider configuration..... | 24 |
| Remove a provider parameter in an existing provider configuration..... | 25 |
| Saving aliases..... | 26 |
| Configuring Kerberos authentication in Apache Knox shared providers..... | 28 |
| Configuring group mapping in Knox..... | 30 |
| | |
| Management of services for Apache Knox through Cloudera Manager..... | 32 |
| Enable proxy for a known service in Apache Knox..... | 33 |
| Disable proxy for a known service in Apache Knox..... | 34 |
| Add custom service to existing descriptor in Apache Knox Proxy..... | 35 |
| Add a custom descriptor to Apache Knox..... | 37 |
| | |
| Management of Service Parameters for Apache Knox via Cloudera | |
| Manager..... | 38 |
| Add custom service parameter to descriptor..... | 38 |
| Modify custom service parameter in descriptor..... | 39 |
| Remove custom service parameter from descriptor..... | 41 |
| | |
| Load balancing for Apache Knox..... | 43 |

| | |
|---|-----------|
| Generate and configure a signing keystore for Knox in HA..... | 44 |
| Knox Gateway token integration..... | 44 |
| Overview..... | 44 |
| Token configurations..... | 46 |
| Generate tokens..... | 51 |
| Manage Knox Gateway tokens..... | 54 |
| Knox Token API..... | 56 |
| Manage Knox metadata..... | 60 |
| Knox SSO Cookie Invalidation..... | 61 |
| Concurrent session verification (Tech Preview)..... | 63 |

Apache Knox overview

Dynamically generating Knox topology files

Topology files can be dynamically generated from combinations of provider configurations and descriptors, which can be defined using Cloudera Manager.

In the early days of Knox, you enabled Knox proxy by editing topology files directly. The topology files consisted of the following things:

- Provider configurations: For example, authentication, federation, authorization, identity assertion, HA, and so on.
- Services: Native Knox services (for example, KNOXTOKEN) or proxied services.

You configured each of the above list in every topology file.

More recently, topology files can be dynamically generated from combinations of provider configurations and descriptors. Additionally, these provider configurations are now shared; you no longer have to specify configurations (for example, authentication provider, identity assertion provider, or authorization provider) for each topology file. You define a provider configuration which can be shared by many descriptors. The following list describes provider configurations, descriptors, and topologies:

- Provider configurations: A named set of providers, for example, authentication, federation, authentication, authorization, identity assertion, and so on. Provider configurations can be shared across descriptors or topologies.
- Descriptors: References the provider configurations to declare the policy (authentication, authorization, identity assertion, and so on) that goes along with proxying that cluster. Descriptors and topologies are 1-to-1.
- Topologies: Dynamically generated based on the descriptors you define.

However, the topologies that are managed by Cloudera Manager should be read-only. Within a Cloudera Manager managed cluster, use Cloudera Manager for creating additional topologies.

Securing access to Hadoop cluster: Apache Knox

The Apache Knox Gateway (“Knox”) is a system to extend the reach of Apache™ Hadoop® services to users outside of a Hadoop cluster without reducing Hadoop Security. Knox also simplifies Hadoop security for users who access the cluster data and execute jobs. The Knox Gateway is designed as a reverse proxy.

Establishing user identity with strong authentication is the basis for secure access in Hadoop. Users need to reliably identify themselves and then have that identity propagated throughout the Hadoop cluster to access cluster resources.

Layers of defense for a CDP cluster

- Authentication: Kerberos

Cloudera uses Kerberos for authentication. Kerberos is an industry standard used to authenticate users and resources within a Hadoop cluster. CDP also includes Cloudera Manager, which simplifies Kerberos setup, configuration, and maintenance.

- Perimeter Level Security: Apache Knox

Apache Knox Gateway is used to help ensure perimeter security for Cloudera customers. With Knox, enterprises can confidently extend the Hadoop REST API to new users without Kerberos complexities, while also maintaining compliance with enterprise security policies. Knox provides a central gateway for Hadoop REST

APIs that have varying degrees of authorization, authentication, SSL, and SSO capabilities to enable a single access point for Hadoop.

Cloudera recommends that you leverage the default PAM Authentication Provider for the benefits in performance and ease of administration rather than direct LDAP. For more details, see *Considerations for Knox*.

- Authorization: Ranger
- OS Security: Data Encryption and HDFS

Related Information

[Considerations for Knox](#)

Apache Knox Gateway overview

A conceptual overview of the Apache Knox Gateway, a reverse proxy.

Overview

Knox integrates with Identity Management and SSO systems used in enterprises and allows identity from these systems be used for access to Hadoop clusters.

Knox Gateway provides security for multiple Hadoop clusters, with these advantages:

- Simplifies access: Extends Hadoop's REST/HTTP services by encapsulating Kerberos to within the Cluster.
- Enhances security: Exposes Hadoop's REST/HTTP services without revealing network details, providing SSL out of the box.
- Centralized control: Enforces REST API security centrally, routing requests to multiple Hadoop clusters.
- Enterprise integration: Supports LDAP, Active Directory, SSO, SAML and other authentication systems.

Typical security flow: Firewall, routed through Knox Gateway

Knox can be used with both unsecured Hadoop clusters, and Kerberos secured clusters. In an enterprise solution that employs Kerberos secured clusters, the Apache Knox Gateway provides an enterprise security solution that:

- Integrates well with enterprise identity management solutions
- Protects the details of the Hadoop cluster deployment (hosts and ports are hidden from end users)
- Simplifies the number of services with which a client needs to interact

Knox Gateway deployment architecture

Users who access Hadoop externally do so either through Knox, via the Apache REST API, or through the Hadoop CLI tools.

Knox Supported Services Matrix

A support matrix showing which services Apache Knox supports for Proxy and SSO, for both Kerberized and Non-Kerberized clusters.

Table 1: Knox Supported Components

| Component | UI Proxy (with SSO) | API Proxy |
|----------------------|---------------------|-----------|
| Atlas API | # | # |
| Atlas UI | # | # |
| Beacon | | |
| Cloudera Manager API | # | # |

| Component | UI Proxy (with SSO) | API Proxy |
|--|---------------------|-----------|
| Cloudera Manager UI | # | |
| Data Analytics Studio (DAS) | # | |
| Druid | | |
| Falcon | | |
| Flink | | |
| HBase REST API(aka WebHBase & Stargate) | | # |
| HBase UI | # | |
| HDFS UI | # | |
| HiveServer2 HTTP JDBC API (HS2 via HTTP) | | # |
| HiveServer2 LLAP JDBC API | | |
| HiveServer2 LLAP UI | | |
| HiveServer2 UI | | |
| Hue | # | |
| Impala HTTP JDBC API | | # |
| Impala UI | # | |
| JobHistory UI | # | |
| JobTracker | | # |
| Kudu UI | # | |
| Livy API + UI | # | # |
| LogSearch | | |
| NameNode | # | # |
| NiFi | # | # |
| NiFi Registry | # | # |
| Oozie API | # | # |
| Oozie UI | # | |
| Ozone | # | |
| Phoenix (aka Avatica) | | # |
| Profiler | # | |
| Ranger API | # | # |
| Ranger UI | # | |
| Yarn ResourceManager API | # | # |
| Schema Registry API + UI | # | # |
| Streams Messaging Manager (SMM) API | # | # |
| Streams Messaging Manager (SMM) UI | # | |
| Solr | # | # |
| Spark3History UI | # | |
| SparkHistory UI | # | |
| Storm | | |
| Storm LogViewer | | |

| Component | UI Proxy (with SSO) | API Proxy |
|-------------|---------------------|-----------|
| Superset | | |
| WebHCat | | |
| WebHDFS | | # |
| YARN UI | # | |
| YARN UI V2 | # | |
| Zeppelin UI | # | |
| Zeppelin WS | # | |

**Note:**

APIs, UIs, and SSO in the Apache Knox project that are not listed above are considered Community Features.

Community Features are developed and tested by the Apache Knox community but are not officially supported by Cloudera. These features are excluded for a variety of reasons, including insufficient reliability or incomplete test case coverage, declaration of non-production readiness by the community at large, and feature deviation from Cloudera best practices. Do not use these features in your production environments.

Knox Topology Management in Cloudera Manager

In CDP Private Cloud, you can manage Apache Knox topologies via Cloudera Manager using `cdp-proxy` and `cdp-proxy-api`.

Shared providers

The Cloudera Manager configurations where the `cdp-proxy` and `cdp-proxy-api` topologies can be managed are:

- Knox Simplified Topology Management - `cdp-proxy`
- Knox Simplified Topology Management - `cdp-proxy-api`

- The SSO authentication provider is used by the UIs using the Knox SSO capabilities, such as the Home Page UI.
- The API authentication provider is used by predefined topologies, such as admin, metadata or `cdp-proxy-api`.
- You can add or modify new or existing shared provider configurations.
- You can save aliases using a new Knox Gateway command.

Services

You can enable or disable known or custom services in Knox proxy via Cloudera Manager.

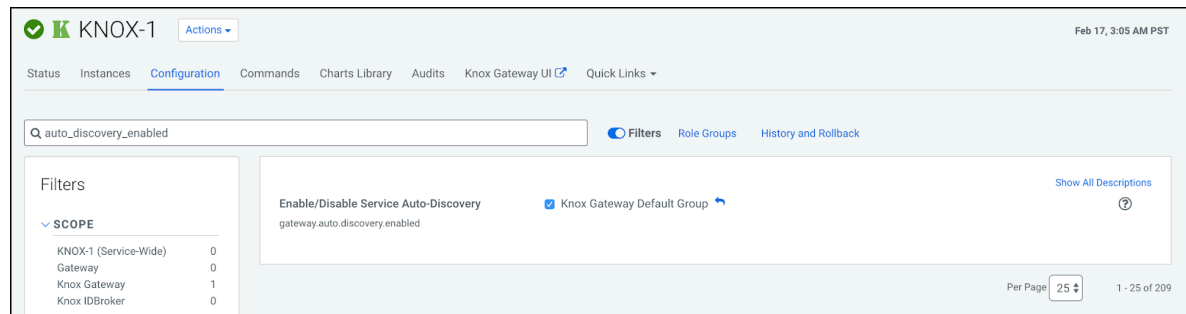
There are two kinds of services in `cdp-proxy`:

- **Known:** officially-supported Knox services. Cloudera Manager provides and manages all the required service definition files.
- **Custom:** unofficial, tech preview, or community feature Knox services. You must supply the service definition files (service.xml and rewrite.xml) that exist in the KNOX_DATA_DIR/services folder. These are not recommended for production environments, and not supported by Cloudera.



Important:

These topologies will be deployed by Cloudera Manager only if Knox's service auto-discovery feature is turned on using the Enable/Disable Service Auto-Discovery checkbox on Cloudera Manager UI:



Important: Adding a custom service will only work if you provide the service definition files (service.xml and rewrite.xml) in the KNOX_DATA_DIR/services folder.

Service parameters

You can add, modify, or remove custom service parameters in Knox proxy via Cloudera Manager.

Considerations for Knox

Learn about the considerations before you get started with Knox.

Default PAM settings for Knox

Secure clusters require local OS accounts. Local OS accounts are most often achieved by using something like SSSD or Centrify which localizes user accounts from user stores or directories including LDAP/AD and so on.

This means that you should be able to use PAM and the local OS accounts straight away as long as they are on the same host as Knox. There are various ways to make local OS accounts available. SSSD or Centrify are technical solutions that make it seem like there are local OS accounts even though they are only in LDAP/AD. You can use real local OS accounts as well.

Proxy Cloudera Manager through Apache Knox

In order to have Cloudera Manager proxied through Knox, there are some steps you must complete.

Procedure

1. Set the value for frontend_url: Cloudera Manager Administration Settings Cloudera Manager Frontend URL :
 - Non-HA value: `https://$Knox_host:$knox_port`
 - HA value: `https://$Knox_loadbalancer_host:$Knox_loadbalancer_port`

2. Set allowed groups, hosts, and users for Knox Proxy: Cloudera Manager Administration Settings External Authentication :
 - Allowed Groups for Knox Proxy: *
 - Allowed Hosts for Knox Proxy: *
 - Allowed Users for Knox Proxy: *
3. Enable Kerberos/SPNEGO authentication for the Admin Console and API: Cloudera Manager Administration Settings External Authentication Enable SPNEGO/Kerberos Authentication for the Admin Console and API: : true
4. From Cloudera Manager Administration Settings External Authentication , set Knox Proxy Principal: Knox.

What to do next

External authentication must be set up correctly. Cloudera Manager must be configured to use LDAP, following the standard procedure for setting up LDAP. This LDAP server should be the same LDAP that populates local users on Knox hosts (if using PAM authentication with Knox), or the same LDAP that Knox is configured to use (if using LDAP authentication with Knox).

However in cases where no LDAP server is available ,corresponding Cloudera Manager local users can be created and assigned roles manually in Cloudera Manager Administrator Users & Roles Add Local User .

Installing Apache Knox

This document provides instructions on how to install Apache Knox using the installation process.

About this task

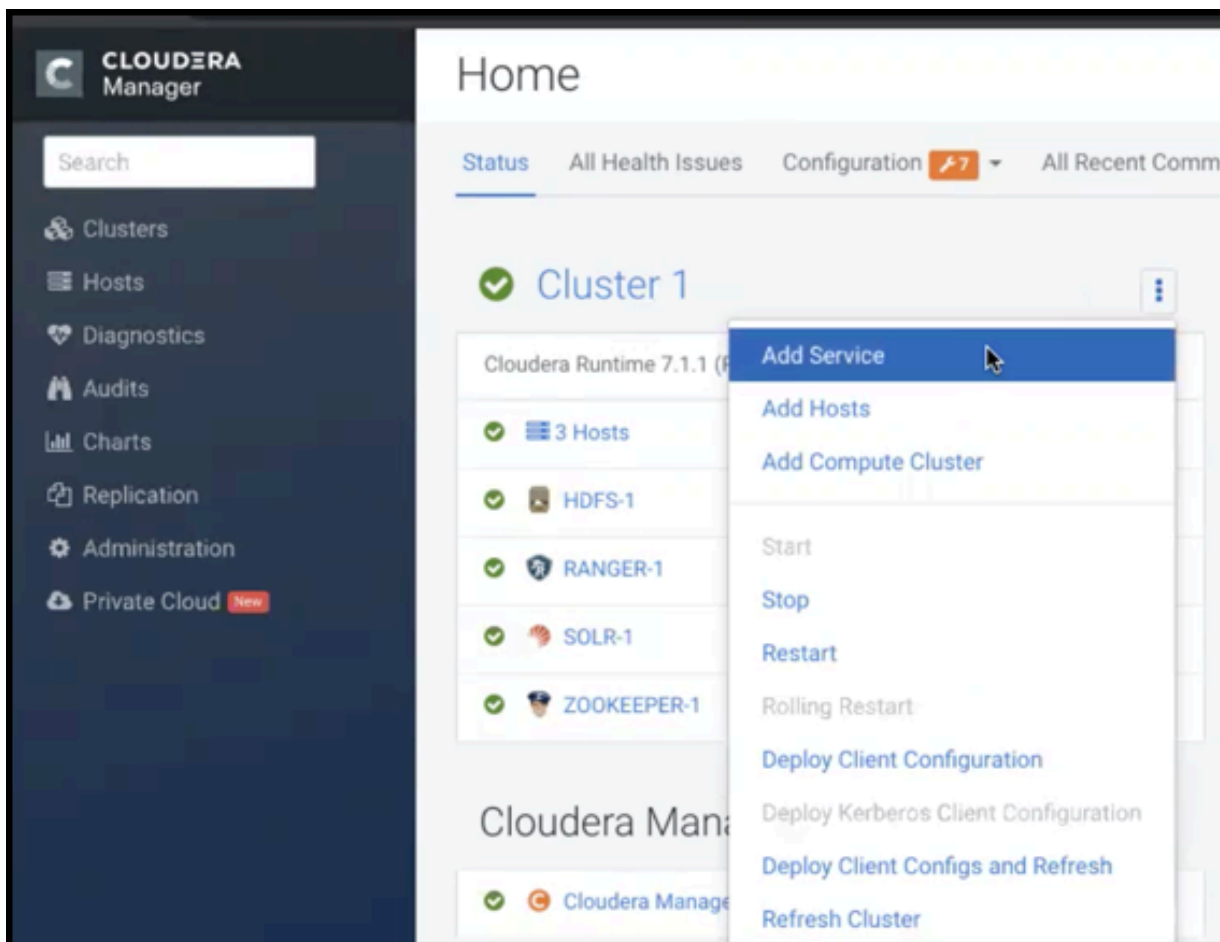
Apache Knox is an application gateway for interacting with the REST APIs and UIs. The Knox Gateway provides a single access point for all REST and HTTP interactions in your Cloudera Data Platform cluster.

Before you begin

When installing Knox, you must have Kerberos enabled on your cluster.

Procedure

1. From your Cloudera Manager homepage, go to Status tab \$Cluster Name ... Add Service



2. From the list of services, select Knox and click Continue.
3. On the **Select Dependencies** page, choose the dependencies you want Knox to set up:

| | |
|--------------------------------------|--|
| HDFS, Ranger, Solr, Zookeeper | For users that require Apache Ranger for authorization. HDFS with Ranger. HDFS depends on Zookeeper, and Ranger depends on Solr. |
| HDFS, Zookeeper | HDFS depends on Zookeeper. |
| No optional dependencies | For users that do not wish to have Knox integrate with HDFS or Ranger. |

4. On the **Assign Roles** page, select role assignments for your dependencies and click Continue:

| Knox service roles | Description | Required? |
|--------------------|--|-----------|
| Knox Gateway | If Knox is installed, at least one instance of this role should be installed. This role represents the Knox Gateway which provides a single access point for all REST and HTTP interactions with Apache Hadoop clusters. | Required |

| Knox service roles | Description | Required? |
|--------------------|---|-----------|
| KnoxIDBroker* | It is strongly recommended that this role is installed on its own dedicated host. As its name suggests this role will allow you to take advantage of Knox's Identity Broker capabilities, an identity federation solution that exchanges cluster authentication for temporary cloud credentials.* | Optional* |
| Gateway | This role comes with the CSD framework. The gateway structure is used to describe the client configuration of the service on each host where the gateway role is installed. | Optional |

* Note: KnoxIDBroker appears in the Assign Roles page, but it is not currently supported in CDP Private Cloud.

5. On the **Review Changes** page, most of the default values are acceptable, but you must Enable Kerberos Authentication and supply the Knox Master Secret. There are additional parameters you can specify or change, listed in "Knox Install Role Parameters".
 - a) Click Enable Kerberos Authentication
 - Kerberos is required where Knox is enabled.
 - b) Supply the Knox Master Secret, e.g. `knoxsecret`.
 - c) Click Continue.
6. The **Command Details** page shows the status of your operation. After completion, your system admin can view logs for your installation under `stdout`.

Apache Knox Install Role Parameters

Reference information on all the parameters available for Knox service roles.

Service-level parameters

Table 2: Required service-level parameters

| Name | In Wizard | Type | Default Value |
|--|-----------|---------|---|
| <code>kerberos.auth.enabled*</code> | Yes | Boolean | false |
| <code>ranger_knox_plugin_hdfs_audit_directory</code> | No | Text | <code>\${ranger_base_audit_url}/knox</code> |
| <code>autorestart_on_stop</code> | No | Boolean | false |
| <code>knox_pam_realm_service</code> | No | Text | login |
| <code>save_alias_command_input_password</code> | No | Text | - |

Knox Gateway role parameters

Table 3: Required parameters for Knox Gateway role

| Name | In Wizard | Type | Default Value |
|------------------------------------|-----------|----------|---|
| <code>gateway_master_secret</code> | Yes | Password | - |
| <code>gateway_conf_dir</code> | Yes | Path | <code>/var/lib/knox/gateway/conf</code> |
| <code>gateway_data_dir</code> | Yes | Path | <code>/var/lib/knox/gateway/data</code> |
| <code>gateway_port</code> | No | Port | 8443 |
| <code>gateway_path</code> | No | Text | gateway |

| Name | In Wizard | Type | Default Value |
|---|-----------|--------|---|
| gateway_heap_size | No | Memory | 1 GB (min = 256 MB; soft min = 512 MB) |
| gateway_ranger_knox_plugin_conf_path | No | Path | /var/lib/knox/ranger-knox-plugin |
| gateway_ranger_knox_plugin_policy_cache_directory | No | Path | /var/lib/ranger/knox/gateway/policy-cache |
| gateway_ranger_knox_plugin_hdfs_audit_spool_directory | No | Path | /var/log/knox/gateway/audit/hdfs/spool |
| gateway_ranger_knox_plugin_solr_audit_spool_directory | No | Path | /var/log/knox/gateway/audit/solr/spool |

Table 4: Optional parameters for Knox Gateway role

| Name | Type | Default Value |
|--|------------|--|
| gateway_default_topology_name | Text | cdp-proxy |
| gateway_auto_discovery_enabled | Boolean | true |
| gateway_cluster_configuration_monitor_interval | Time | 60 seconds (minimum = 30 seconds) |
| gateway_auto_discovery_advanced_configuration_monitor_interval | Time | 10 seconds (minimum = 5 seconds) |
| gateway_cloudera_manager_descriptors_monitor_interval | Time | 10 seconds (minimum = 5 seconds) |
| gateway_auto_discovery_cdp_proxy_enabled_* | Boolean | true |
| gateway_auto_discovery_cdp_proxy_api_enabled_* | Boolean | true |
| gateway_descriptor_cdp_proxy | Text Array | Contains the required properties of cdp-proxy topology |
| gateway_descriptor_cdp_proxy_api | Text Array | Contains the required properties of cdp-proxy-api topology |
| gateway_sso_authentication_provider | Text Array | Contains the required properties of the authentication provider used by the UIs using the Knox SSO capabilities (such as Home Page UI). Defaults to PAM authentication. |
| gateway_api_authentication_provider | Text Array | Contains the required properties of the authentication provider used by pre-defined topologies such as admin, metadata or cdp-proxy-api. Defaults to PAM authentication. |

Management of Knox shared providers in Cloudera Manager

Information on CDP Private Cloud topology management for Knox from within Cloudera Manager.

- Modifying the SSO authentication provider used by the UIs using the Knox SSO capabilities, such as the Home Page UI.
- Modifying the API authentication provider used by predefined topologies, such as admin, metadata or cdp-proxy-api.
- Adding/modifying new/existing shared provider configurations.
- Saving aliases using a new Knox Gateway command.

Configure Apache Knox authentication for PAM

Knox authentication configurations for PAM in Cloudera Manager. PAM is the default SSO authentication provider in CDP Private Cloud.



Note: The knox user needs permission on /etc/shadow for PAM authentication. Allow the knox user to read the /etc/shadow file:

```
groupadd shadow
usermod -a -G shadow knox
chgrp shadow /etc/shadow
chmod g+r /etc/shadow
```

SSO authentication for PAM

In CDP Private Cloud, Cloudera Manager added a new Knox configuration, called Knox Simplified Topology Management - SSO Authentication Provider, with the following initial configuration:

```
role=authentication
authentication.name=ShiroProvider
authentication.param.sessionTimeout=30
authentication.param.redirectToUrl=${GATEWAY_PATH}/knoxssso/knoxauth/login.html
authentication.param.restrictedCookies=rememberme,WWW-Authenticate
authentication.param.urls./*=authcBasic
authentication.param.main.pamRealm=org.apache.knox.gateway.shirorealm.KnoxPamRealm
authentication.param.main.pamRealm.service=login
```

The screenshot shows the Cloudera Manager configuration interface for 'Cluster 1' and 'KNOX-1'. The 'Configuration' tab is active, showing the 'SSO Authentication Provider' configuration. The configuration is organized into two main sections: 'Knox Simplified Topology Management - SSO Authentication Provider' and 'Knox Gateway Default Group'. The configuration parameters are as follows:

| Parameter | Value |
|--|---|
| role | authentication |
| authentication.name | ShiroProvider |
| authentication.param.sessionTimeout | 30 |
| authentication.param.redirectToUrl | \${GATEWAY_PATH}/knoxssso/knoxauth/login.html |
| authentication.param.restrictedCookies | rememberme,WWW-Authenticate |
| authentication.param.main.pamRealm | org.apache.knox.gateway.shirorealm.KnoxPamRealm |
| authentication.param.main.pamRealm.service | login |
| authentication.param.urls./* | authcBasic |

The interface also includes a 'Filters' sidebar on the left, a search bar at the top, and a 'Show All Descriptions' link on the right.

Every change here is applied to the knoxssso topology that affects manager, homepage and cdp-proxy topologies as they are using the federation provider.

API authentication for PAM

A new Knox configuration has been added for CDP Private Cloud, called Knox Simplified Topology Management - API Authentication Provider, with the following initial configuration:

```
role=authentication
```

```

authentication.name=ShiroProvider
authentication.param.sessionTimeout=30
authentication.param.urls./**=authcBasic
authentication.param.main.pamRealm=org.apache.knox.gateway.shirorealm.KnoxPamRealm
authentication.param.main.pamRealm.service=login

```

Every change here is applied to the admin, metadata, and cdp-proxy-api topologies.

Configure Apache Knox authentication for AD/LDAP

Knox authentication configurations for LDAP and AD in Cloudera Manager.

SSO authentication for AD/LDAP

In the following sample you will see how to change the PAM authentication (which comes default with Knox) to LDAP authentication. It is as simple as removing the default PAM related configuration in ShiroProvider and add LDAP related properties (e.g. with demo LDAP server configuration):

```

role=authentication
authentication.name=ShiroProvider
authentication.param.sessionTimeout=30
authentication.param.redirectToUrl=${GATEWAY_PATH}/knoxssso/knoxauth/login.html
authentication.param.restrictedCookies=rememberme,WWW-Authenticate
authentication.param.urls./**=authcBasic
authentication.param.main.ldapRealm=org.apache.knox.gateway.shirorealm.KnoxLdapRealm
authentication.param.main.ldapContextFactory=org.apache.knox.gateway.shirorealm.KnoxLdapContextFactory
authentication.param.main.ldapRealm.contextFactory=$ldapContextFactory
authentication.param.main.ldapRealm.contextFactory.authenticationMechanism=simple
authentication.param.main.ldapRealm.contextFactory.url=ldap://localhost:33389
authentication.param.main.ldapRealm.contextFactory.systemUsername=uid=guest,ou=people,dc=hadoop,dc=apache,dc=org
authentication.param.main.ldapRealm.contextFactory.systemPassword=${ALIAS=knoxLdapSystemPassword}
authentication.param.main.ldapRealm.userSearchBase=DC=EXAMPLE,DC=COM
authentication.param.main.ldapRealm.userSearchAttributeName=sAMAccountName
authentication.param.main.ldapRealm.userObjectClass=person
authentication.param.main.ldapRealm.groupSearchBase=OU=Groups,DC=EXAMPLE,DC=COM
authentication.param.main.ldapRealm.groupObjectClass=group
authentication.param.remove=main.pamRealm
authentication.param.remove=main.pamRealm.service

```

After you finished editing the properties you have to save the configuration changes. This will make the Refresh Needed stale configuration indicator appear. Once the cluster refresh finishes, all topologies that are configured to use Knox SSO will be authenticated by the configured LDAP server.

The screenshot shows the Cloudera Manager interface for configuring a Knox Gateway Default Group. The left sidebar contains filters for SCOPE, CATEGORY, and STATUS. The main area displays a list of configuration parameters for the 'Knox Gateway Default Group'.

| Filter Category | Filter Name | Count | |
|-----------------|-----------------------|-------------|---|
| SCOPE | KNOX-1 (Service-Wide) | 0 | |
| | Gateway | 0 | |
| | Knox Gateway | 1 | |
| | Knox IDBroker | 0 | |
| CATEGORY | Advanced | 0 | |
| | Logs | 0 | |
| | Main | 1 | |
| | Monitoring | 0 | |
| | Performance | 0 | |
| | Ports and Addresses | 0 | |
| | Resource Management | 0 | |
| | Security | 0 | |
| | Stacks Collection | 0 | |
| | STATUS | Error | 0 |
| | | Warning | 0 |
| | | Edited | 0 |
| | | Non-default | 1 |
| Has Overrides | | 0 | |
| | | | |
| | | | |

The main configuration area shows the following parameters for the 'Knox Gateway Default Group':

- role=authentication
- authentication.name=ShiroProvider
- authentication.param.sessionTimeout=30
- authentication.param.redirectToUrl=/\${GATEWAY_PATH}/knoxssso/knoxauth/login.html
- authentication.param.restrictedCookies=rememberme,WWW-Authenticate
- authentication.param.urls./**=authcBasic
- authentication.param.main.ldapRealm=org.apache.knox.gateway.shirorealm.KnoxLdapRealm
- authentication.param.main.ldapContextFactory=org.apache.knox.gateway.shirorealm.KnoxLdapContextFactory
- authentication.param.main.ldapRealm.contextFactory=\${ldapContextFactory}
- authentication.param.main.ldapRealm.contextFactory.authenticationMechanism=simple
- authentication.param.main.ldapRealm.contextFactory.url=ldap://localhost:33389
- authentication.param.main.ldapRealm.contextFactory.systemUsername=uid=guest,ou=people,dc=org
- authentication.param.main.ldapRealm.contextFactory.systemPassword=\${ALIAS=knoxLdapSystemPassword}
- authentication.param.main.ldapRealm.userDnTemplate=uid={0},ou=people,dc=hadoop,dc=apache,dc=org
- authentication.param.remove=main,pamRealm
- authentication.param.remove=main,pamRealm.service



Note:

As you can see we used a Knox alias when we declared the system password instead of writing the plain text password there. To make it easier for the end-users a new Knox Gateway command was created that allows them to save aliases on all hosts where a Knox Gateway is running. See [Saving aliases](#).

To verify:

```
$ curl -ku <username>:<password> 'https://johndoe-1.abc.cloudera.com:8443/gateway/admin/api/v1/providerconfig/knoxssso'
...
}, {
  "role" : "authentication",
  "name" : "ShiroProvider",
  "enabled" : true,
  "params" : {
    "main.ldapContextFactory" : "org.apache.knox.gateway.shirorealm.KnoxLdapContextFactory",
    "main.ldapRealm" : "org.apache.hadoop.gateway.shirorealm.KnoxLdapRealm",
    "main.ldapRealm.contextFactory" : "${ldapContextFactory}",
    "main.ldapRealm.contextFactory.authenticationMechanism" : "simple",
    "main.ldapRealm.contextFactory.systemPassword" : "${ALIAS=knoxLdapSystemPassword}",
    "main.ldapRealm.contextFactory.systemUsername" : "uid=guest,ou=people,dc=hadoop,dc=apache,dc=org",
    "main.ldapRealm.contextFactory.url" : "ldap://localhost:33389",
    "main.ldapRealm.userDnTemplate" : "uid={0},ou=people,dc=hadoop,dc=apache,dc=org",
    "redirectToUrl" : "/${GATEWAY_PATH}/knoxssso/knoxauth/login.html",
    "restrictedCookies" : "rememberme,WWW-Authenticate",
    "sessionTimeout" : "30",
    "urls./**" : "authcBasic"
  }
}
```




Note: Any change in SSO authentication configuration alters the Knox SSO topology. This affects the manager, homepage, and cdp-proxy topologies because the SSO cookie federation provider is used.

API authentication for AD/LDAP

In the following sample you will see how to change the PAM authentication (which comes default with Knox) to LDAP authentication:

```
role=authentication
authentication.name=ShiroProvider
authentication.param.sessionTimeout=30
authentication.param.urls./**=authcBasic
authentication.param.main.ldapRealm=org.apache.knox.gateway.shirorealm.KnoxLdapRealm
authentication.param.main.ldapContextFactory=org.apache.knox.gateway.shirorealm.KnoxLdapContextFactory
authentication.param.main.ldapRealm.contextFactory=$ldapContextFactory
authentication.param.main.ldapRealm.contextFactory.authenticationMechanism=simple
authentication.param.main.ldapRealm.contextFactory.url=ldap://localhost:33389
authentication.param.main.ldapRealm.contextFactory.systemUsername=uid=guest,ou=people,dc=hadoop,dc=apache,dc=org
authentication.param.main.ldapRealm.contextFactory.systemPassword=${ALIAS=knoxLdapSystemPassword}
authentication.param.main.ldapRealm.userSearchBase=DC=EXAMPLE,DC=COM
authentication.param.main.ldapRealm.userSearchAttributeName=sAMAccountName
authentication.param.main.ldapRealm.userObjectClass=person
authentication.param.main.ldapRealm.groupSearchBase=OU=Groups,DC=EXAMPLE,DC=COM
authentication.param.main.ldapRealm.groupObjectClass=group
authentication.param.remove=main.pamRealm
authentication.param.remove=main.pamRealm.service
```

Every change here goes directly into admin, metadata, and cdp-proxy-api topologies.

Knox CLI testing tools

Learn how to use the Knox Command Line Interface (CLI) to run diagnostic tests.

The Knox CLI is a command line utility that can be used to manage and test various aspects of a Knox deployment.

You must set the following environment variables before using the Knox CLI:

```
export KNOX_GATEWAY_DATA_DIR="/var/lib/knox/gateway/data"
export KNOX_GATEWAY_CONF_DIR="/var/lib/knox/gateway/conf"
```

The `knoxcli.sh` command line utility script is located in the `/opt/cloudera/parcels/CDH/lib/knox/bin` directory.

Invoke the CLI by using the following command:

```
/opt/cloudera/parcels/CDH/lib/knox/bin/knoxcli.sh
```

Knox CLI LDAP authentication and authorization testing

You can use the following command format to authenticate a user name and password against LDAP.

```
bin/knoxcli.sh user-auth-test [--cluster c] [--u username] [--p password] [-g] [--d] [--help]
```

This command tests the ability of a topology to connect, authenticate, and authorize a user with an LDAP server. The only required argument is the `--cluster` argument to specify the name of the topology you want to use. The topology must be valid (passes a `validate-topology` command). If the `-u` and `-p` arguments are not specified, you are prompted for a user name and password.

If authentication is successful, the command attempts to use the topology to do an LDAP group lookup. The topology must be configured correctly to do this. If the topology is not configured correctly, groups are not returned and no errors are printed unless the `--g` argument is specified. Currently, this command only works if a topology supports the use of ShiroProvider for authentication.

Table 5: LDAP authentication and authorization arguments

| Argument | Description | Required? |
|------------------------|--|-----------|
| <code>--cluster</code> | The name of the cluster to authenticate. | Yes |
| <code>--u</code> | The user name to authenticate with. | No |
| <code>--p</code> | The password to authenticate with. | No |
| <code>--g</code> | Specifies that you want to return a user's groups. If not specified, group lookup errors will not be returned. | No |
| <code>--d</code> | Print extra debug information for a failed authentication. | No |

Configure Apache Knox Authentication for SAML

Knox authentication configurations for SAML in Cloudera Manager. Knox uses `pac4j` provider for SAML.

Configuring SAML with the `pac4j` provider

You can configure SAML in Cloudera Manager through Knox Configuration Knox Simplified Topology Management - SSO Authentication Provider. An example minimal configuration is shown below:

```
authentication.enabled=false
role=federation
federation.name=pac4j
federation.param.clientName=SAML2Client federation.param.pac4j.callbackUrl
=https://knox.example.com:8443/gateway/knoxssso/api/v1/webssso federation.param.
saml.identityProviderMetadataPath=/etc/knox/conf/idp.xml federation.param.
saml.serviceProviderMetadataPath=/etc/knox/conf/sp.xml
federation.param.saml.serviceProviderEntityId=knox-sp-entity
authentication.param.remove=main.pamRealm
authentication.param.remove=main.pamRealm.service
```

You can find additional advanced configuration options in the upstream Apache Knox and `pac4j` documentation.

You can obtain the Identity Provider (IdP) metadata that Knox needs from your IdP admins. The information required to configure the SAML Identity Provider, including the Knox entity id and AssertionConsumerService endpoint, is contained in the SAML Service Provider (SP) metadata which Knox generates automatically. Once the SP metadata has been written to the `serviceProviderMetadataPath` on the Knox host, you can send it to the IdP admins to complete the configuration at the IdP side.

Add a new shared provider configuration

Provider configurations are definitions of authentication and authorization controls for services proxied by Knox, which may be referenced by one or more descriptors.

Procedure

1. Define the providers:

- a) From Cloudera Manager Knox Configuration , add a new entry in Knox Gateway Advanced Configuration Snippet (Safety Valve) for `conf/cdp-resources.xml_role_safety_valve`.
- b) Name the provider configuration, and specify the desired providers and their corresponding configuration attributes.
Provider configuration entries are named as `providerConfigs:TOPOLOGY_NAME` (E.G., `providerConfigs:myTopology`).
- c) For each provider, the role is declared (E.G., `role=authentication`, `role=authorization`), and subsequently configured by defining properties of that role (E.G., `authentication.name=ShiroProvider`, `authentication.param.sessionTimeout=30`).

Example (LDAP authentication and Ranger authorization)

- Name=`providerConfigs:ldap-ranger-provider`
- Value=

```
role=authentication#
authentication.name=ShiroProvider#
authentication.param.sessionTimeout=30#
authentication.param.main.ldapRealm=org.apache.hadoop.gateway.shirorealm.KnoxLdapRealm#
authentication.param.main.ldapContextFactory=org.apache.knox.gateway.shirorealm.KnoxLdapContextFactory#
authentication.param.main.ldapRealm.contextFactory=$ldapContextFactory#
authentication.param.main.ldapRealm.contextFactory.authenticationMechanism=simple#
authentication.param.main.ldapRealm.contextFactory.url=ldap://ldap-host:33389#
authentication.param.main.ldapRealm.contextFactory.systemUsername=uid=guest,ou=people,dc=hadoop,dc=apache,dc=org#
authentication.param.main.ldapRealm.userDnTemplate=uid={0},ou=people,dc=hadoop,dc=apache,dc=org#
authentication.param.urls./**=authcBasic#
role=authorization#
authorization.name=XASecurePDPKnox
```

2. Save your changes.
3. Refresh your cluster: the Refresh needed stale configuration indicator appears; click it and wait until the refresh process completes.
4. Validate using the Knox homepage.

Verify that your topology is generated with the services and URLs you specified.

TLS Mutual Authentication

Mutual authentication with TLS provides the Knox gateway with the means to establish a strong trust relationship with another party. This is especially useful when applications that act on behalf of end-users send requests to Knox.

While this feature does establish an authenticated trust relationship with the client application, it does not determine the end-user identity through this authentication. It will continue to look for credentials or tokens that represent the end-user within the request and authenticate or federate the identity accordingly.

To enable TLS Mutual Authentication, set the following in CM Knox Configuration Knox Service (or Gateway) Advanced Configuration Snippet (Safety Valve) for `conf/gateway-site.xml`:

```
gateway.client.auth.needed = true
```

The truststore path for client authentication can be set in Cloudera Manager Knox Configuration Knox Service (or Gateway) Advanced Configuration Snippet (Safety Valve) for `conf/gateway-site.xml`

```
gateway.truststore.path
```

This parameter can point to the standard truststore (typically `truststore.jks`) on the host if it contains all of the necessary `TrustedCertEntry`'s for client authentication. Even if the JKS is password protected, Knox can still get `TrustedCertEntry` content from it, so no password is necessary.

If `gateway.client.auth.needed = true` and `gateway.truststore.path` is unset, then it will look at this default location `/var/lib/knox/gateway/data/security/keystores/gateway.jks` for truststore AND keystore entries, which is an atypical configuration for JKS usage in our stack and not recommended.

These two parameters are distinct, but can point to the same truststore: `gateway.truststore.path` is for client authentication in the context of TLS Mutual Authentication, and `gateway.httpClient.truststore.path` is for when the Knox Gateway is an HTTP Client to other TLS servers.

Management of existing Apache Knox shared providers

You can add, modify, or disable an existing shared provider configuration in Apache Knox via Cloudera Manager.

The following default shared provider configurations are deployed in CDP Private Cloud with Knox:

Table 6: Default shared provider configurations

| Configuration | Used by these topologies |
|---------------|----------------------------------|
| admin | admin |
| homepage | homepage |
| knoxsso | homepage cdp-proxy manager |
| manager | manager |
| metadata | metadata |
| pam | cdp-proxy-api |
| sso | cdp-proxy |



Note: `pam` and `sso` are available only if service auto-discovery is enabled for Knox Gateway role.

The following changes are allowed in any of these shared providers:

- Disable a particular provider
- Modify a particular provider
- Add a new provider

All of these actions can be done via editing the Knox Gateway Advanced Configuration Snippet (Safety Valve) for `conf/cdp-descriptors.xml` by implementing the following language elements:

- The key of a new entry should be like this: `providerConfigs: providerConfig_1 [,providerConfig_2,...,providerConfig_3]`
- The value should contain the following name/value pairs separated by a hash (#) character:

```
role=webappsec|authentication|federation|identity-assertion|authorization|
hostmap|ha
$role.name=ROLE_NAME (e.g. ShiroProvider)
```

```
$role.enabled=true|false (optional; defaults to 'true')
$role.param.param_1=value_1 (parameters are optional too)
...
$role.param_N.param1=value_N
```

Add a new provider in an existing provider configuration

An example of how to add a new provider to the authorization provider in the manager shared provider configuration.

About this task

In this example you will see how to add a new HA provider (this time only the ATLAS service will be configured for high availability) in the manager shared provider configuration . This particular authorization provider is set as follows (in its JSON descriptor):

```
{
  "role": "authorization",
  "name": "AclsAuthz",
  "enabled": "true",
  "params": {
    "knox.acl.mode": "OR",
    "knox.acl": "KNOX_ADMIN_USERS;KNOX_ADMIN_GROUPS;* "
  }
}
```

Procedure

- From Cloudera Manager Knox Configuration, add the following entry in the Knox Gateway Advanced Configuration Snippet (Safety Valve) for `conf/cdp-descriptors.xml`:

- name = `providerConfigs:manager`
- value = `role=authorization#authorization.name=AclsAuthz#authorization.enabled=false#authorization.param.knox.acl=myTestUser;KNOX_ADMIN_GROUPS;*#authorization.param.knox.acl.mode=OR#role=ha#ha.name=HaProvider#ha.param.ATLAS=enabled=true;maxFailoverAttempts=3;failoverSleep=1000;maxRet`

The screenshot shows the Cloudera Manager interface for Knox Configuration. The main area displays the configuration form for a new provider. The form includes fields for Name, Value, and Description. The Name field contains `providerConfigs:manager`. The Value field contains `role=authorization#authorization.name=AclsAuthz#authorization.enabled=false#`. The Description field is empty. There is a checkbox for 'Final' which is unchecked. The interface also shows a 'Filters' sidebar on the left and a 'Knox Gateway Default Group' dropdown set to 'Knox Gateway Default Group'.

The screenshot shows the Cloudera Manager interface for Knox Configuration, specifically the XML editor. The main area displays the XML configuration for the provider. The XML content is as follows:

```
<property>
<name>providerConfigs:manager</name>
<value>role=authorization#authorization.name=AclsAuthz#authorization.enabled=false#authorization.param.knox.acl=myTestUser;KNOX_ADMIN_GROUPS;*#authorization.param.knox.acl.mode=OR#role=ha#ha.name=HaProvider#ha.param.ATLAS=enabled=true;maxFailoverAttempts=3;failoverSleep=1000;maxRetryAttempts=300;retrySleep=1000</value>
</property>
```

The interface also shows a 'Filters' sidebar on the left and a 'Knox Gateway Default Group' dropdown set to 'Knox Gateway Default Group'.

- Save your changes.
- Refresh the cluster.
- Validate:

```
$ curl -ku <username>:<password> 'https://johndoe-1.abc.cloudera.com:8443/gateway/admin/api/v1/providerconfig/manager'
{
  "providers" : [
    ...
  ], {
    "role" : "authorization",
    "name" : "AclsAuthz",
    "enabled" : false,
    "params" : {
      "knox.acl" : "myTestUser;KNOX_ADMIN_GROUPS;*",
      "knox.acl.mode" : "OR"
    }
  }, {
    "role" : "ha",
    "name" : "HaProvider",
```

```

    "enabled" : true,
    "params" : {
      "ATLAS" : "enabled=true;maxFailoverAttempts=3;failoverSleep=1000;maxRetryAttempts=300;retrySleep=1000"
    }
  }
}

```

Modify a provider in an existing provider configuration

An example of how to modify the authorization provider in the manager shared provider configuration.

About this task

In this example you will see how to modify the authorization provider in the manager shared provider configuration. This particular authorization provider is set as follows (in its JSON descriptor):

```

{
  "role": "authorization",
  "name": "AclsAuthz",
  "enabled": "true",
  "params": {
    "knox.acl.mode": "OR",
    "knox.acl": "KNOX_ADMIN_USERS;KNOX_ADMIN_GROUPS;*"
  }
}

```

Procedure

- From Cloudera Manager Knox Configuration, add the following entry in the Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml:
 - name = providerConfigs:manager
 - value = role=authorization#authorization.name=AclsAuthz#authorization.enabled=false#authorization.param.knox.acl=myTestUser;KNOX_ADMIN_GROUPS;*#authorization.param.knox.acl.mode=OR

The screenshot shows the Cloudera Manager interface for Knox Gateway configuration. The main content area displays a configuration snippet for 'providerConfigs:manager' with the following value: 'role=authorization#authorization.name=AclsAuthz#authorization.enabled=false#authorization.param.knox.acl=myTestUser;KNOX_ADMIN_GROUPS;*#authorization.param.knox.acl.mode=OR'. The interface includes a search bar, filters, and a 'Final' checkbox.

With this change you are authorizing a user called myTestUser to login and execute administrative actions on the Knox Admin API.

- Save your changes.
- Refresh the cluster.
- Validate:

```

$ curl -ku <username>:<password> 'https://johndoe-1.abc.cloudera.com:8443/gateway/admin/api/v1/providerconfig/manager'
{

```

```

"providers" : [
  ...
  ], {
    "role" : "authorization",
    "name" : "AclsAuthz",
    "enabled" : false,
    "params" : {
      "knox.acl" : "myTestUser;KNOX_ADMIN_GROUPS;*",
      "knox.acl.mode" : "OR"
    }
  }, {
    "role" : "ha",
    "name" : "HaProvider",
    "enabled" : true,
    "params" : {
      "ATLAS" : "enabled=true;maxFailoverAttempts=3;failoverSleep=1000;maxRetryAttempts=300;retrySleep=1000"
    }
  } ]
}

```

Disable a provider in an existing provider configuration

An example of how to disable the authorization provider in the manager shared provider configuration.

About this task

In this example you will see how to disable the authorization provider in the manager shared provider configuration. This particular authorization provider is set as follows (in its JSON descriptor):

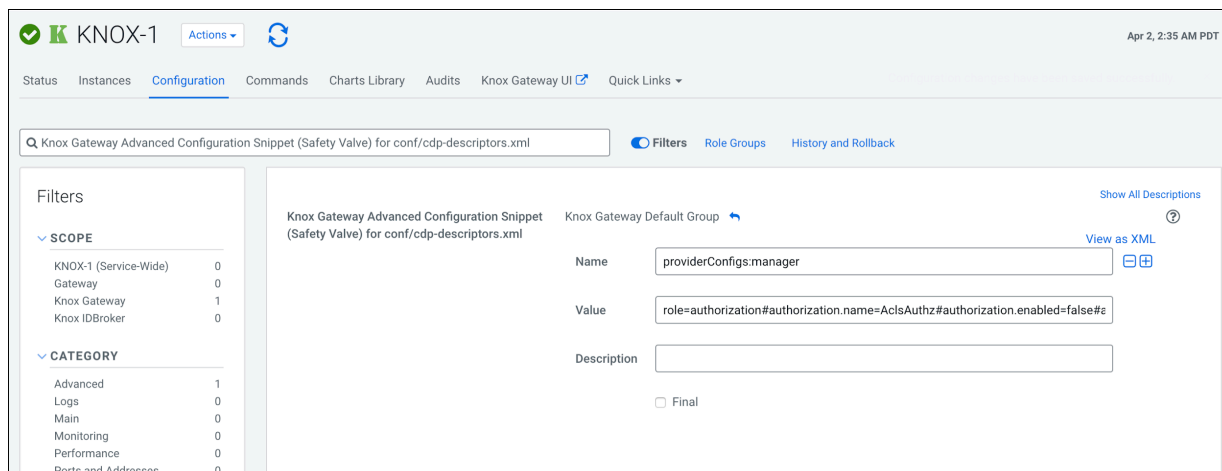
```

{
  "role": "authorization",
  "name": "AclsAuthz",
  "enabled": "true",
  "params": {
    "knox.acl.mode": "OR",
    "knox.acl": "KNOX_ADMIN_USERS;KNOX_ADMIN_GROUPS;*"
  }
}

```


Procedure

- From Cloudera Manager Knox Configuration, add the following entry in the Knox Gateway Advanced Configuration Snippet (Safety Valve) for `conf/cdp-descriptors.xml`:
 - `name = providerConfigs:manager`
 - `value = role=authorization#authorization.name=AclsAuthz#authorization.enabled=false#authorization.param.knox.acl=KNOX_ADMIN_USERS;KNOX_ADMIN_GROUPS;*#authorization.param.knox.acl.mode=OR`



- Save your changes.
- Refresh the cluster.
- Validate:

```
$ curl -ku <username>:<password> 'https://johndoe-1.abc.cloudera.com:8443/gateway/admin/api/v1/providerconfig/manager'
{
  "providers" : [
    . . .
  ], {
    "role" : "authorization",
    "name" : "AclsAuthz",
    "enabled" : false,
    "params" : {
      "knox.acl" : "myTestUser;KNOX_ADMIN_GROUPS;*",
      "knox.acl.mode" : "OR"
    }
  }, {
    "role" : "ha",
    "name" : "HaProvider",
    "enabled" : true,
    "params" : {
      "ATLAS" : "enabled=true;maxFailoverAttempts=3;failoverSleep=1000;maxRetryAttempts=300;retrySleep=1000"
    }
  } ]
}
```

What to do next

The only change is that the enabled flag was changed to false.

Remove a provider parameter in an existing provider configuration

An example of how to remove the authentication parameter from a shared provider configuration.

About this task

In this example you will see how to remove an authentication provider parameter in the pam shared provider configuration. This particular provider is set as follows:

```
{
  "providers" : [ {
    "role" : "authentication",
    "name" : "ShiroProvider",
    "enabled" : true,
    "params" : {
      "main.pamRealm" : "org.apache.knox.gateway.shirorealm.KnoxPamRealm",
      "main.pamRealm.service" : "login",
      "sessionTimeout" : "30"
    }
  } ],
  "readOnly" : true
}
```

Procedure

1. From Cloudera Manager Knox Configuration, add the following entry in the Knox Gateway Advanced Configuration Snippet (Safety Valve) for `conf/cdp-descriptors.xml`:
 - name = providerConfigs:pam
 - value = role=authentication#authentication.name=ShiroProvider#authentication.param.remove=sessionTimeout#authentication.param.main.pamRealm=org.apache.knox.gateway.shirorealm.KnoxPamRealm#authentication.param.main.pamRealm.service=login
2. Save your changes.
3. Refresh the cluster.
4. Validate:

```
$ curl -ku <username>:<password> 'https://johndoe-1.abc.cloudera.com:8443/gateway/admin/api/v1/providerconfig/pam' {
  "providers" : [ {
    "role" : "authentication",
    "name" : "ShiroProvider",
    "enabled" : true,
    "params" : {
      "main.pamRealm" : "org.apache.knox.gateway.shirorealm.KnoxPamRealm",
      "main.pamRealm.service" : "login"
    }
  } ],
  "readOnly" : true
}
```

Saving aliases

There is a new command available for the Knox Gateway role which allows end-users to save an alias=password pair to an arbitrary number of topologies on each host where an instance of the Knox Gateway is installed without the need of running the Knox CLI tool manually.

A new password-type input field is added, called `save_alias_command_input_password`. The format of an entry in this input field should be: `topology_name_1[:topology_name_2:...:topology_name_N].alias_name=password`

Example: `cdp-proxy-api:admin:metadata.knoxLdapSystemPassword=guest-password`.

After the end-users entered a meaningful and valid value and saved the configuration changes they can run the command from Knox's action list: Actions/Save Alias.



Tip: If you need to add a Gateway level alias, please use `__gateway` as topology name. For instance: `__gateway.knoxLdapSystemPassword=admin-password`.

Cluster 1

KNOX-1 Actions

Status Instances **Configuration** Commands Charts Library Audits Knox Gateway UI Quick Links

Save Alias

Filters Role Groups History and Rollback

Save Alias Command Input

save_alias_command_input_password

KNOX-1 (Service-Wide)

.....

Per Page 25 1 - 25 of 216

Configuration changes have been saved successfully.

Cluster 1

KNOX-1 Actions

Status **Instances** Configuration Commands Charts Library Audits Web UI Quick Links

Search

Filters

STATUS **Good Health** 3

COMMISSION STATE

MAINTENANCE MODE

RACK ID

Actions for Selected

Add Role Instances Role Groups

| Status | Role Type | State | Hostname | Commission State | Role Group |
|--------------------------|-----------------------------|---------|--------------------|------------------|----------------------------|
| <input type="checkbox"/> | ✓ Knox Gateway | Started |@cloudera.com | Commissioned | Knox Gateway Default Group |
| <input type="checkbox"/> | ✓ Knox Gateway | Started |@cloudera.com | Commissioned | Knox Gateway Default Group |
| <input type="checkbox"/> | ✓ Knox Gateway | Started |@cloudera.com | Commissioned | Knox Gateway Default Group |

Last Updated: Apr 2, 3:05:30 AM PDT

1 - 3 of 3

Cluster 1

KNOX-1 Actions

Status **Instances** Configuration

Health Tests

✓ Knox Gateway Health

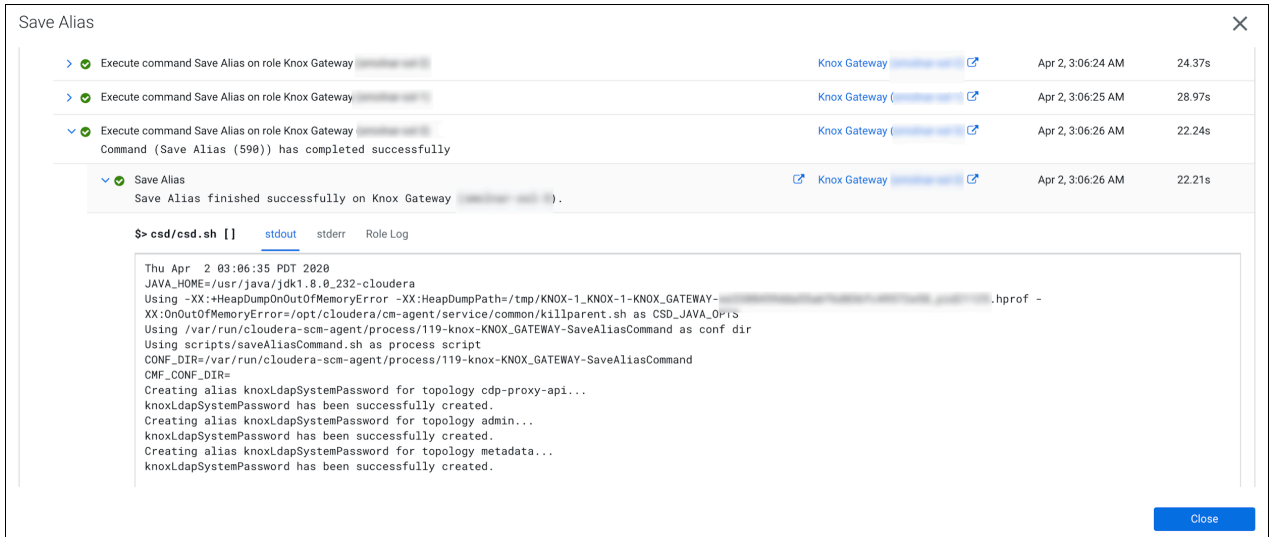
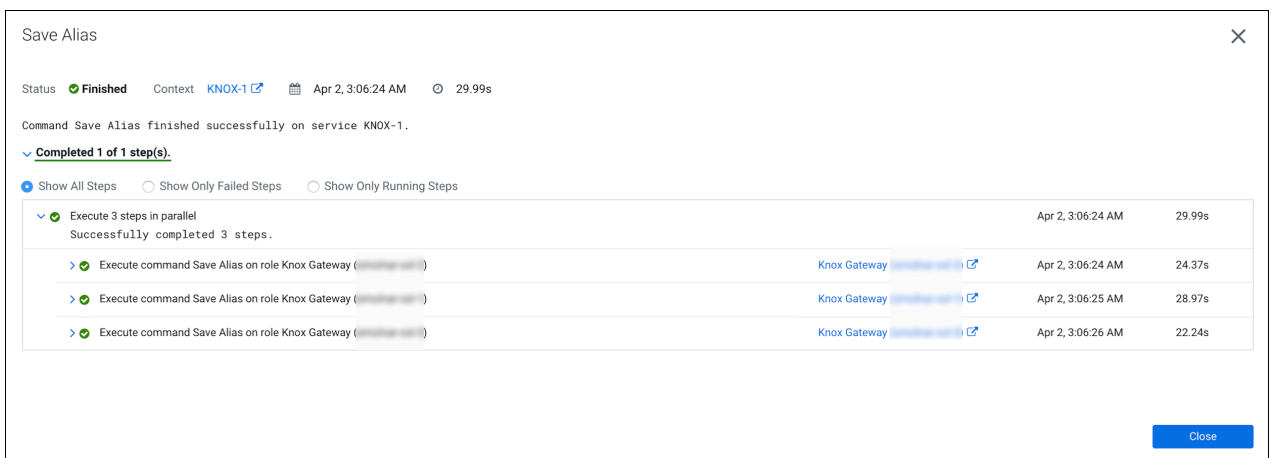
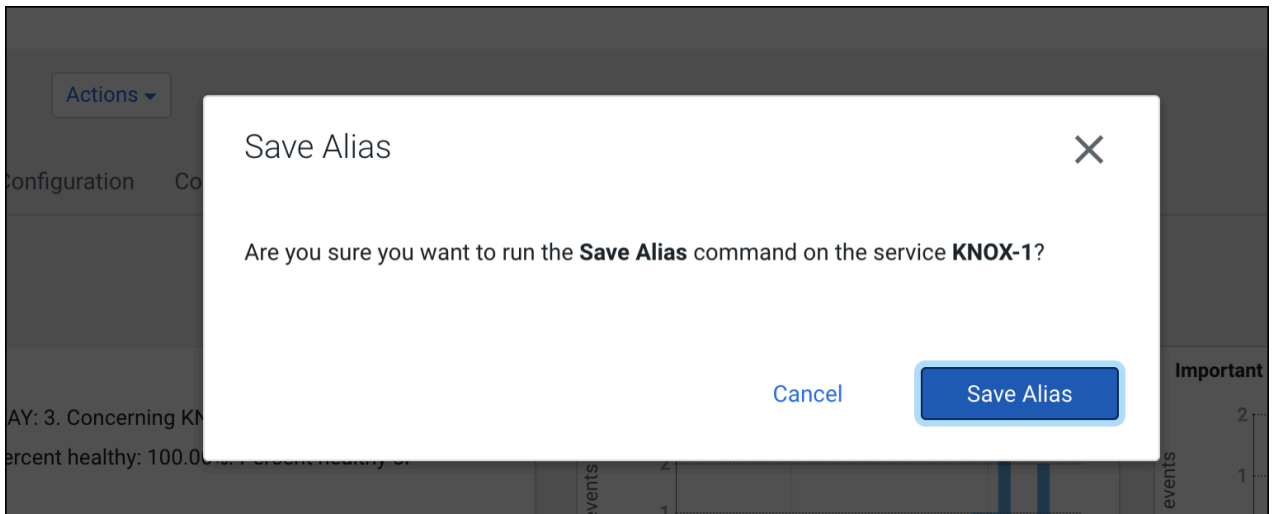
Start

Restart

Rolling Restart

Save Alias

Stop



Configuring Kerberos authentication in Apache Knox shared providers

An example of how to add the kerberos-auth configuration provider from Cloudera Manager.

Procedure

1. From Cloudera Manager Knox Configuration, add the following entry in the Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml:

- Name = providerConfigs:kerberos-providers
- Value =

```
role=authentication#
authentication.name=HadoopAuth#
authentication.param.sessionTimeout=30#
authentication.param.config.prefix=hadoop.auth.config#
authentication.param.hadoop.auth.config.type=kerberos#
authentication.param.hadoop.auth.config.signature.secret=${ALIAS=AUTH_CONFIG_SIGNATURE_SECRET}
authentication.param.hadoop.auth.config.token.validity=1800#
authentication.param.hadoop.auth.config.cookie.path=/#
authentication.param.hadoop.auth.config.simple.anonymous.allowed=false#
authentication.param.hadoop.auth.config.kerberos.principal=AUTH_CONFIG_KERBEROS_PRINCIPAL#
authentication.param.hadoop.auth.config.kerberos.keytab=AUTH_CONFIG_KERBEROS_KEYTAB#
authentication.param.hadoop.auth.config.kerberos.name.rules=DEFAULT
```



Important: Paste the value as a single line, without line-breaks.

2. Add a safety valve name/value pair in Cloudera Manager Knox Configuration, in Knox Gateway Environment Advanced Configuration Snippet (Safety Valve):

```
Name = IDBROKER_KERBEROS_DT_PROXYUSER_BLOCK
Value = "proxyuser_block": "none"
```

| Knox Gateway Environment Advanced Configuration Snippet (Safety Valve) | Knox Gateway Default Group | View as Text |
|--|----------------------------|--------------------------------------|
| KNOX_GATEWAY_role_env_safety_valve | Key | IDBROKER_KERBEROS_DT_PROXYUSER_BLOCK |
| | Value | "proxyuser_block": "none" |

3. Save your changes.
4. Refresh the cluster.
5. Validate with a curl command: `curl -k https://host-10-00-100-100:8443/gateway/admin/api/v1/providerconfig/kerberos-providers`.

```
# curl -k https://host-10-00-100-100:8443/gateway/admin/api/v1/providerconfig/kerberos-providers
{
  "providers" : [ {
    "role" : "authentication",
    "name" : "HadoopAuth",
    "enabled" : true,
    "params" : {
      "config.prefix" : "hadoop.auth.config",
      "hadoop.auth.config.kerberos.keytab" : "/var/run/cloudera-scm-agent/process/81-knox-KNOX_GATEWAY/knox.keytab",
      "hadoop.auth.config.kerberos.name.rules" : "DEFAULT",
      "hadoop.auth.config.kerberos.principal" : "HTTP/host-10-00-100-100.coe.cloudera.com@CLOUDERA.COM",
      "hadoop.auth.config.signature.secret" : "${ALIAS=AUTH_CONFIG_SIGNATURE_SECRET}" ,
    }
  }
]
```

```

    "hadoop.auth.config.simple.anonymous.allowed" : "false",
    "hadoop.auth.config.token.validity" : "1800",
    "hadoop.auth.config.type" : "kerberos",
    "proxyuser_block" : "none"
  }, {
    "role" : "identity-assertion",
    "name" : "HadoopGroupProvider",
    "enabled" : true,
    "params" : {
      "CENTRAL_GROUP_CONFIG_PREFIX" : "gateway.group.config."
    }
  }, {
    "role" : "authorization",
    "name" : "XASecurePDPKnox",
    "enabled" : true,
    "params" : { }
  }, {
    "role" : "ha",
    "name" : "HaProvider",
    "enabled" : true,
    "params" : {
      "HBASE" : "maxFailoverAttempts=3;failoverSleep=1000;enabled=true",
      "HIVE" : "maxFailoverAttempts=3;failoverSleep=1000;enabled=true;zoo
keeperEnsemble=maxFailoverAttempts=3;failoverSleep=1000;enabled=true;zoo
keeperEnsemble=gbl20175161.systems.uk.company:2181,gbl20175162.systems.u
k.company:2181,gbl20175163.systems.uk.company:2181;zookeeperNamespace=hi
veserver2",
      "OOZIE" : "maxFailoverAttempts=3;failoverSleep=1000;enabled=true",
      "WEBHCAT" : "maxFailoverAttempts=3;failoverSleep=1000;enabled=true",
      "WEBHDFS" : "maxFailoverAttempts=3;failoverSleep=1000;maxRetryAtte
mpts=300;retrySleep=1000;enabled=true"
    }
  } ],
  "readOnly" : true
}

```

Related Information

[Saving aliases](#)

Configuring group mapping in Knox

Learn how to use HadoopGroupProvider to configure group mapping.

About this task

The role for this provider is identity-assertion and name is HadoopGroupProvider:

```

<provider>
  <role>identity-assertion</role>
  <name>HadoopGroupProvider</name>
  <enabled>true</enabled>
  <<param> ... </param>
</provider>

```

All the configurations for HadoopGroupProvider reside in the provider section of a gateway topology file. The `hadoop.security.group.mapping` property determines the implementation. Some of the valid implementations are as follows:

- `org.apache.hadoop.security.JniBasedUnixGroupsMappingWithFallback`

This is the default implementation and is picked up if `hadoop.security.group.mapping` is not specified. This implementation determines if the Java Native Interface (JNI) is available. If JNI is available, the implementation uses the API to resolve a list of groups for a user. If JNI is not available then the shell implementation, `org.apache.hadoop.security.ShellBasedUnixGroupsMapping` is used. It shells out with the `bash -c groups` command (for a Linux/Unix environment) or the `net group` command (for a Windows environment) to resolve a list of groups for a user.

- `org.apache.hadoop.security.LdapGroupsMapping`

This implementation connects directly to an LDAP server to resolve the list of groups. However, this should only be used if the required groups reside exclusively in LDAP, and are not materialized on the Unix servers.

To enable group lookup using identity assertion as `HadoopGroupProvider`, perform the following steps:

Procedure

1. Go to Cloudera Manager Knox Configuration .
2. Search for the Knox Gateway Advanced Configuration Snippet (Safety Valve) for `conf/cdp-resources.xml` property, and add the following entry:

```
# name = providerConfigs:sso
# value = role=federation#federation.name=SSOCookieProvider#federation.
param.sso.authentication.provider.url=
https://<knox_hostname>:8443/gateway/knoxssso/api/v1/websso#role=identity-
assertion#identity-assertio...
HadoopGroupProvider#identity-assertion.enabled=true#identity-assertion.pa
ram.CENTRAL_GROUP_CONFIG_PREFIX=gateway.group.
config#role=authorization#authorization.name=XASecurePDPKnox#authorizatio
n.enabled=true
```



Note: Replace your Knox hostname in `<knox_hostname>`.

The following is a sample demo of LDAP configurations for the LDAP setting:

```
<param>
<name>gateway.group.config.hadoop.security.group.mapping</name>
<value>org.apache.hadoop.security.LdapGroupsMapping</value>
</param>
<param>
<name>gateway.group.config.hadoop.security.group.mapping.ldap.bind.user</
name>
<value>uid=tom,ou=people,dc=hadoop,dc=apache,dc=org</value>
</param>
<param>
<name>gateway.group.config.hadoop.security.group.mapping.ldap.bind.pass
word</name>
<value>tom-password</value>
</param>
<param>
<name>gateway.group.config.hadoop.security.group.mapping.ldap.url</name>
<value>ldap://localhost:33389</value>
</param>
<param>
<name>gateway.group.config.hadoop.security.group.mapping.ldap.base</name>
<value></value>
</param>
<param>
<name>gateway.group.config.hadoop.security.group.mapping.ldap.search.fil
ter.user</name>
```

```

<value>(&amp;(|(objectclass=person)(objectclass=applicationProcess))(cn
={0}))</value>
</param>
<param>
<name>gateway.group.config.hadoop.security.group.mapping.ldap.search.filter.group</name>
<value>(objectclass=groupOfNames)</value>
</param>
<param>
<name>gateway.group.config.hadoop.security.group.mapping.ldap.search.attr.member</name>
<value>member</value>
</param>
<param>
<name>gateway.group.config.hadoop.security.group.mapping.ldap.search.attr.group.name</name>
<value>cn</value>
</param>
</provider>

```

3. Save your changes.
4. Refresh the cluster.

Management of services for Apache Knox through Cloudera Manager

You can enable or disable known or custom services in Knox proxy through Cloudera Manager.

There are two kinds of services in cdp-proxy:

- **Known:** Officially-supported Knox services. Cloudera Manager provides and manages all the required service definition files.
- **Custom:** Unofficial, tech preview, or community feature Knox services. You must supply the service definition files (service.xml and rewrite.xml) exist in the \$CDP_PARCEL_DIR/lib/knox/data/services folder. These are not recommended for production environments, and not supported by Cloudera.



Note: If you upgrade to a newer version of CDP, the previously added custom service definitions might disappear from the new location. Therefore, you must ensure that those files are there after upgrading to a newer CDP version.



Important:

These topologies will be deployed by Cloudera Manager only if Knox's service auto-discovery feature is turned on using the Enable/Disable Service Auto-Discovery checkbox on Cloudera Manager UI:

For a comprehensive list of known services that can be enabled, see “Knox Supported Services Matrix”.

Related Information

[Knox Supported Services Matrix](#)

Enable proxy for a known service in Apache Knox

How to enable auto-discovery for a known service in Knox proxy via Cloudera Manager.

About this task

“Known” services are officially-supported Knox services (like Apache Atlas, Ranger, Solr, etc.) Cloudera Manager provides and manages all the required service definition files.

For the purposes of this example, we add ATLAS and ATLAS UI to cdp-proxy. You can add more services; for a comprehensive list of knoxn services that can be enabled, see “Knox Supported Services Matrix”.

Procedure

1. From Cloudera Manager Knox Configuration, check the Gateway Auto Discovery (cdp-proxy) - \$Component boxes.

In this example, we enable:

- gateway_auto_discovery_cdp_proxy_enabled_atlas
- gateway_auto_discovery_cdp_proxy_enabled_atlas_ui

The screenshot shows the Cloudera Manager interface for KNOX-1 configuration. The search bar contains the text 'gateway_auto_discovery_cdp_proxy_enabled_atlas'. The configuration table displays two entries:

| Configuration Name | Enabled | Role Group |
|--|-------------------------------------|----------------------------|
| Enable Auto Discovery (cdp-proxy) - Atlas API | <input checked="" type="checkbox"/> | Knox Gateway Default Group |
| Enable Auto Discovery (cdp-proxy) - Atlas Web UI | <input checked="" type="checkbox"/> | Knox Gateway Default Group |

2. Save your changes.
3. The ‘Refresh needed’ stale configuration indicator appears; click it and wait until the refresh process finishes.

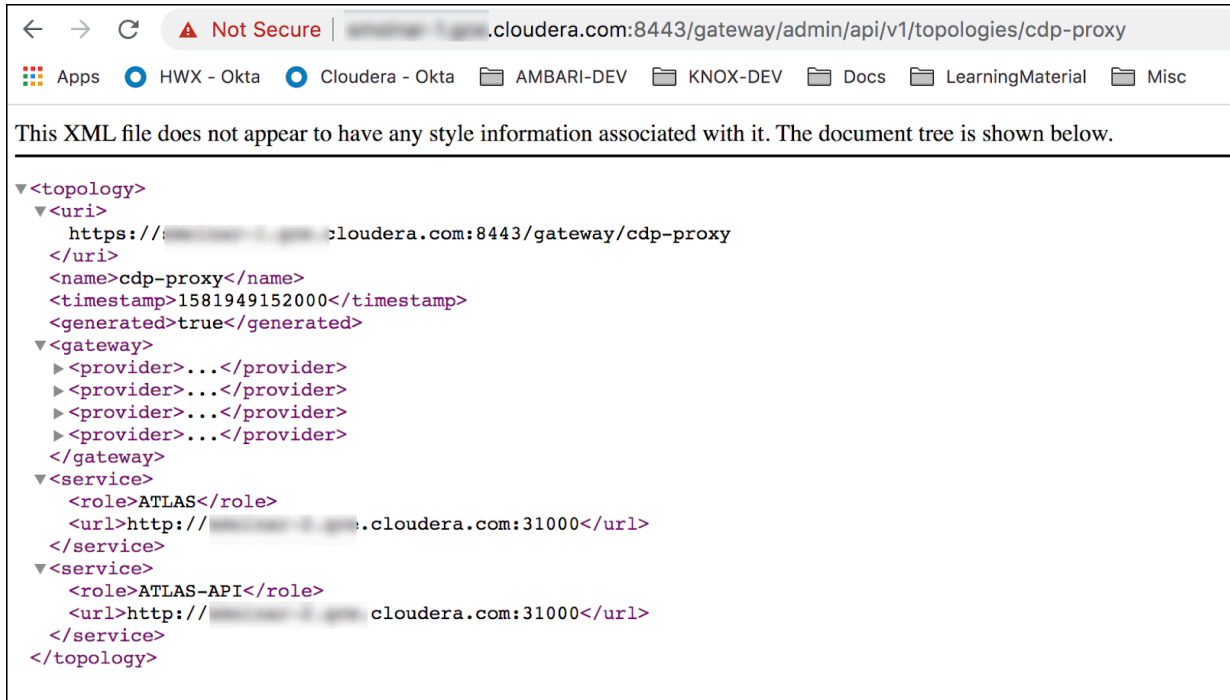
The screenshot shows the 'Stale Configurations' page in Cloudera Manager. The file 'conf/auto-discovery-advanced-configuration-cdp-proxy.properties' is open, showing a diff view of the configuration. The diff highlights the following changes:

```

@@ -1,6 +1,6 @@
1 -gateway.auto.discovery.cdp.proxy.enabled.atlas=false
2 -gateway.auto.discovery.cdp.proxy.enabled.atlas-api=false
1 +gateway.auto.discovery.cdp.proxy.enabled.atlas=true
2 +gateway.auto.discovery.cdp.proxy.enabled.atlas-api=true
3 gateway.auto.discovery.cdp.proxy.enabled.cm-api=false
4 gateway.auto.discovery.cdp.proxy.enabled.cm-ui=false
5 gateway.auto.discovery.cdp.proxy.enabled.hbaseui=false
6 gateway.auto.discovery.cdp.proxy.enabled.hdfsui=false

```

4. Validate that ATLAS in cdp-proxy was added by going to the following URL: `https://$KNOX_GATEWAY_HOST:$PORT/$GATEWAY_PATH/admin/api/v1/topologies/cdp-proxy`.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<topology>
  <uri>https://cloudera.com:8443/gateway/cdp-proxy</uri>
  <name>cdp-proxy</name>
  <timestamp>1581949152000</timestamp>
  <generated>true</generated>
  <gateway>
    <provider>...</provider>
    <provider>...</provider>
    <provider>...</provider>
    <provider>...</provider>
  </gateway>
  <service>
    <role>ATLAS</role>
    <url>http://cloudera.com:31000</url>
  </service>
  <service>
    <role>ATLAS-API</role>
    <url>http://cloudera.com:31000</url>
  </service>
</topology>

```

Related Information

[Add custom service parameter to descriptor](#)

[Knox Supported Services Matrix](#)

Disable proxy for a known service in Apache Knox

How to remove auto-discovery for a known service in Knox proxy via Cloudera Manager.

About this task

“Known” services are officially-supported Knox services (like Apache Atlas, Ranger, Solr, etc.) Cloudera Manager provides and manages all the required service definition files.

In this example, we are going to remove the previously added ATLAS and ATLAS-UI services from cdp-proxy. We disable the `gateway_auto_discovery_cdp_proxy_enabled_atlas` and `gateway_auto_discovery_cdp_proxy_enabled_atlas_ui` checkboxes on Knox’s Configuration page in CM, save the changes and refresh the cluster.

Procedure

- From Cloudera Manager Knox Configuration, uncheck the Gateway Auto Discovery (cdp-proxy) - \$Component boxes.

In this example, we disable:

- gateway_auto_discovery_cdp_proxy_enabled_atlas
- gateway_auto_discovery_cdp_proxy_enabled_atlas_ui

- Save your changes.
- The 'Refresh needed' stale configuration indicator appears; click it and wait until the refresh process finishes.

- Validate that custom service got removed by going to the following URL: `http://$KNOX_GATEWAY_HOST:$PORT/$GATEWAY_PATH/admin/api/v1/topologies/cdp-proxy`.

Add custom service to existing descriptor in Apache Knox Proxy

How to add a custom service to an existing descriptor in Knox proxy using Cloudera Manager.

About this task

“Custom” services are unofficial, tech preview, or community feature Knox services. You must supply the service definition files (service.xml and rewrite.xml) which exist in the `KNOX_DATA_DIR/services` folder. These are not recommended for production environments, and not supported by Cloudera.

In this example, a custom service (`MY_SERVICE`) is added in `cdp-proxy` with the following attributes:

- Version : the service’s version, for example, 1.0.0.
- URL: the service URL, for example, `https://sampleHost:1234`.
- Service parameter: a sample service parameter, for example, `myValue`.



Important: Adding a custom service only works if you provide the service definition files (service.xml and rewrite.xml) in the `KNOX_DATA_DIR/services` folder.

To achieve the goals you need to add three new entries with the above-listed parameters in Knox Simplified Topology Management - `cdp-proxy`. Then you save the changes, refresh the cluster and check if the newly added custom service is available in `cdp-proxy`.

Procedure

1. From Cloudera Manager Knox Configuration , add the three new entries with the above-listed parameters.

```
MY_SERVICE:version=1.0.0
MY_SERVICE:url=https://sampleHost:1234
MY_SERVICE:customServiceParameter=myValue
```

2. Save your changes.
3. The ‘Refresh needed’ stale configuration indicator appears; click it and wait until the refresh process completes.

4. Validate that MY_SERVICE in cdp-proxy is added by navigating to the following URL: `http://$KNOX_GATEWAY_HOST:$PORT/$GATEWAY_PATH/admin/api/v1/topologies/cdp-proxy`.

Add a custom descriptor to Apache Knox

How to add a custom descriptor to Apache Knox using Cloudera Manager.

About this task

Custom descriptors can be deployed to Apache Knox using Cloudera Manager. These descriptors, combined with referenced provider configurations, are transformed into Knox topologies. Using Cloudera Manager means that these descriptors only ever need to be changed in one place to affect all Knox Gateway instances in the cluster.

Fundamentally, descriptors contain the declaration of services to proxy and a reference to provider configuration defining how authentication and authorization for those proxied services should be handled. A descriptor also may similarly declare Knox applications as topologies do.

Service declarations consist of at least the name of the service being proxied. They optionally include one or more endpoint URLs and one or more service-specific parameters.

Descriptors optionally include discovery information, allowing Knox to dynamically discover the endpoint URLs for the declared services.

Procedure

1. Define the descriptor contents:

- a) From Cloudera Manager Knox Configuration, add a new entry in Knox Gateway Advanced Configuration Snippet (Safety Valve) for `conf/cdp-resources.xml_role_safety_valve`.
- b) Name the topology, specify the providerConfigRef, and enumerate the services and associated service URLs. Optional service details include version (E.G., `HIVE:version=0.13.0`) and service parameters (E.G., `HIVE:httpclient.connectionTimeout=5m`)

Static URL Example (HIVE and WEBHDFS with PAM authentication)

- Name=`my-custom-topology`
- Value=

```
providerConfigRef=pam#
HIVE:url=https://hive-host-1:10001/cliservice#
WEBHDFS:url=https://hdfs-host-1:20470/webhdfs#
WEBHDFS:url=https://hdfs-host-2:20470/webhdfs
```

Discovery Example (HIVE and WEBHDFS with PAM authentication)



Note: If the CDP cluster is not enabled with Auto-TLS, then you must add the Cloudera Manager certificate to the Knox truststore and restart the Knox service.

- Name=`my-discoverable-topology`
- Value=

```
discoveryType=ClouderaManager#
discoveryAddress=https://cm-host:7183#
cluster=Cluster 1#
providerConfigRef=pam#
HIVE#
WEBHDFS
```

2. Save the changes.

- Refresh the Knox instances' configuration: the Refresh needed stale configuration indicator appears; click it and wait until the refresh process completes.
- Validate using the Knox homepage.
Verify that your topology is generated with the services and URLs you specified.

Management of Service Parameters for Apache Knox via Cloudera Manager

You can add, modify, or remove custom service parameters in Knox proxy via Cloudera Manager.

Add custom service parameter to descriptor

How to add a custom service parameter to a descriptor using Cloudera Manager.

Before you begin

The descriptor you wish to add a custom service parameter to must be enabled. See “Add a known service to cdp-proxy”.

About this task

In this example, you are adding a custom service parameter with a custom value (myCustomServiceParameter=myValue) to ATLAS in cdp-proxy.

Procedure

- From Cloudera Manager Knox Configuration, add a new line in the Knox Simplified Topology Management - cdp-proxy panel in the following format:
`$SERVICE_NAME[:$PARAMETER_NAME=$PARAMETER_VALUE].`
 ATLAS:myCustomServiceParameter=myValue

The url and version parameter names are preserved keywords to set the given service's URL and version. Valid declarations:

```
ATLAS:url=http://localhost:123
ATLAS:version:3.0.0
ATLAS:test.parameter.name=test.parameter.value
```

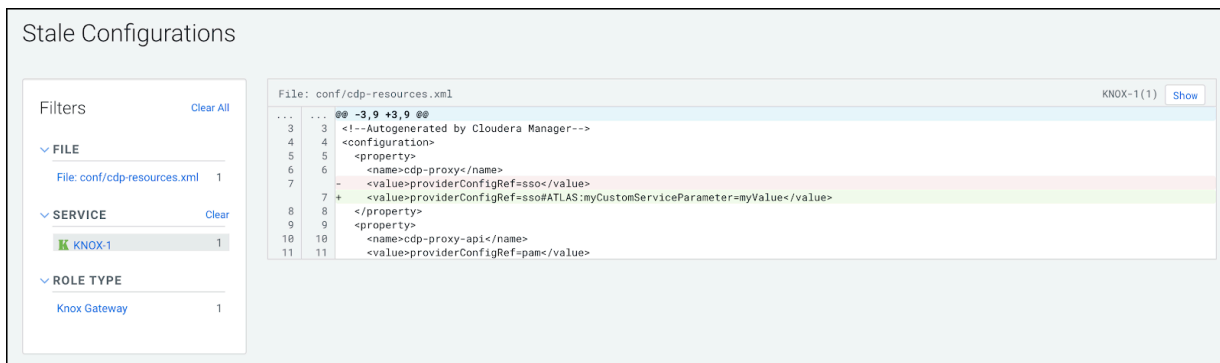
The screenshot shows the Cloudera Manager interface for configuring the Knox service. The main panel displays the configuration for the 'Knox Gateway Default Group' under the 'Knox Simplified Topology Management - cdp-proxy' descriptor. The configuration includes the following parameters:

- `providerConfigRef=sso`
- `ATLAS:myCustomServiceParameter=myValue`
- `providerConfigRef=pam`

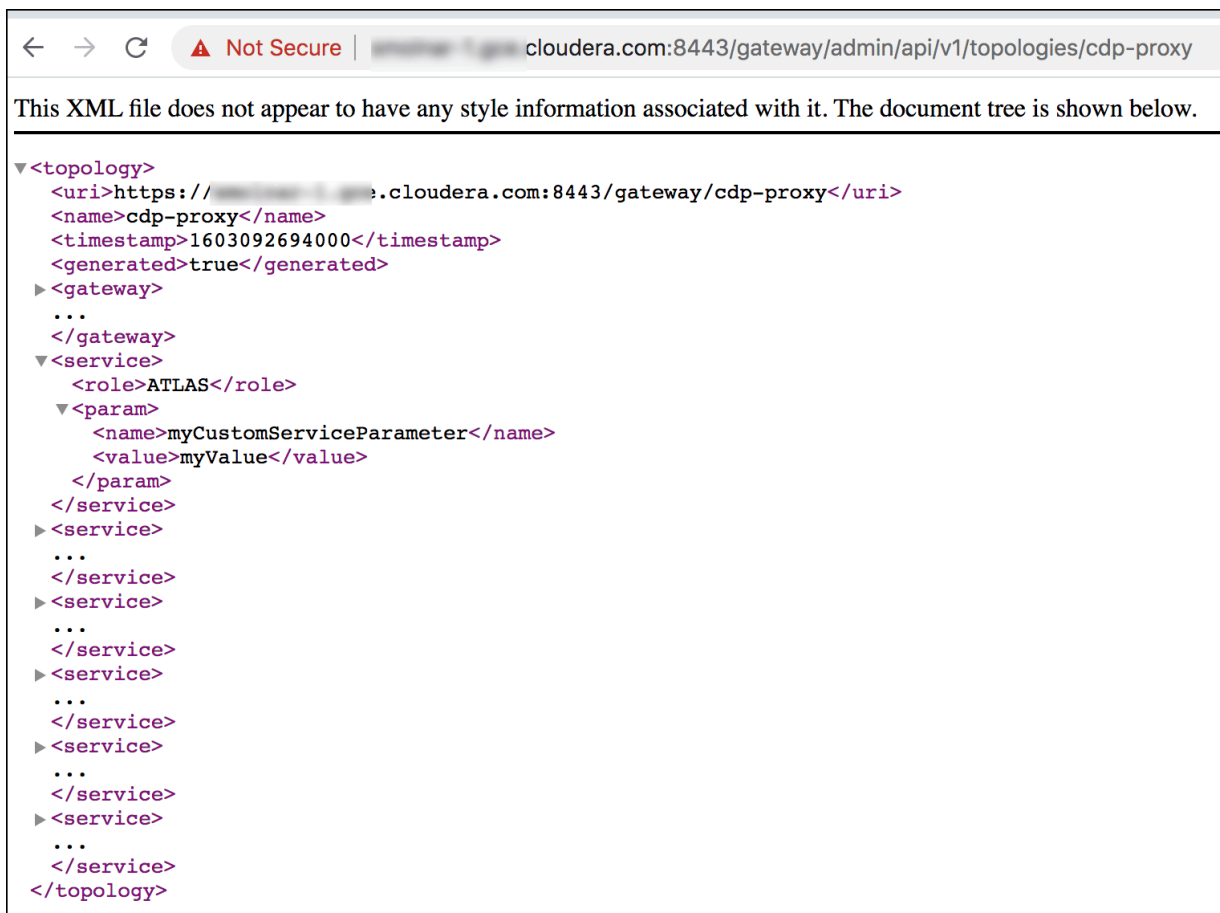
The interface also shows a search bar for 'Knox descriptor block', a filters sidebar, and a 'Show All Descriptions' link. The bottom right corner indicates 'Per Page 25' and '1 - 25 of 220'.

- Save your changes.

- The 'Refresh needed' stale configuration indicator appears; click it and wait until the refresh process completes.



- Validate that ATLAS in cdp-proxy got updated with the new service parameter by navigating to the following URL: `https://$KNOX_GATEWAY_HOST:$PORT/$GATEWAY_PATH/admin/api/v1/topologies/cdp-proxy`.



Modify custom service parameter in descriptor

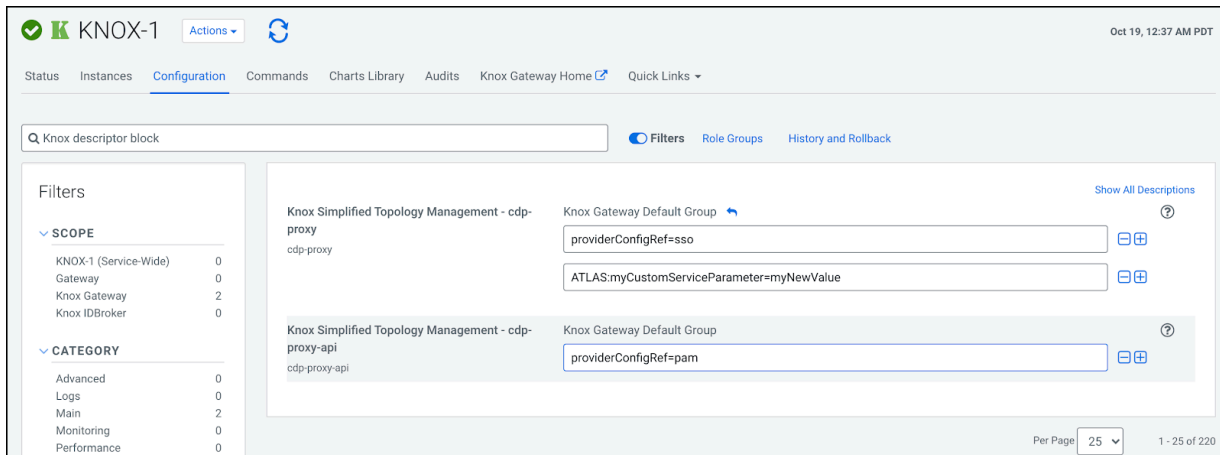
How to edit a custom service parameter in a Knox descriptor using Cloudera Manager.

About this task

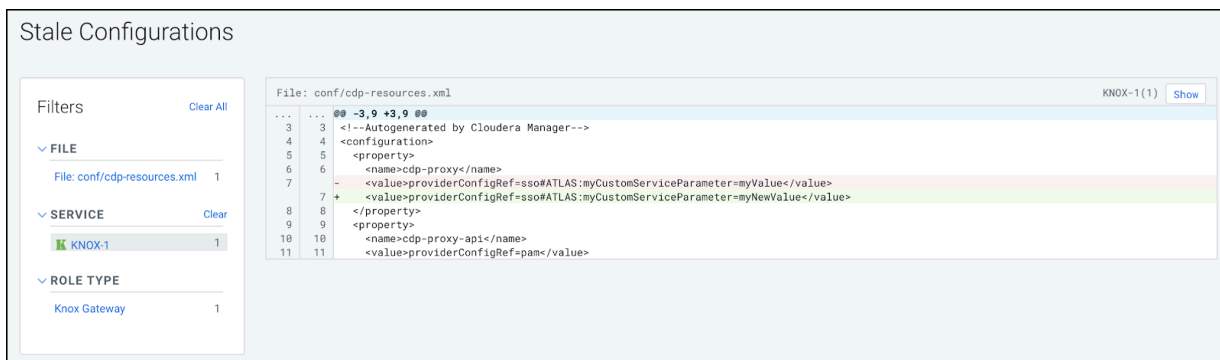
In this sample, we are going to update a previously entered service parameter - `myCustomServiceParameter=myValue` to `myNewValue`- for ATLAS in cdp-proxy. We change that entry, save our changes, and refresh our cluster.

Procedure

1. From Cloudera Manager Knox Configuration , change the service parameter in the Knox Simplified Topology Management - cdp-proxy panel. Change ATLAS:myCustomServiceParameter=myValue to Atlas:myCustomServiceParameter=myNewValue

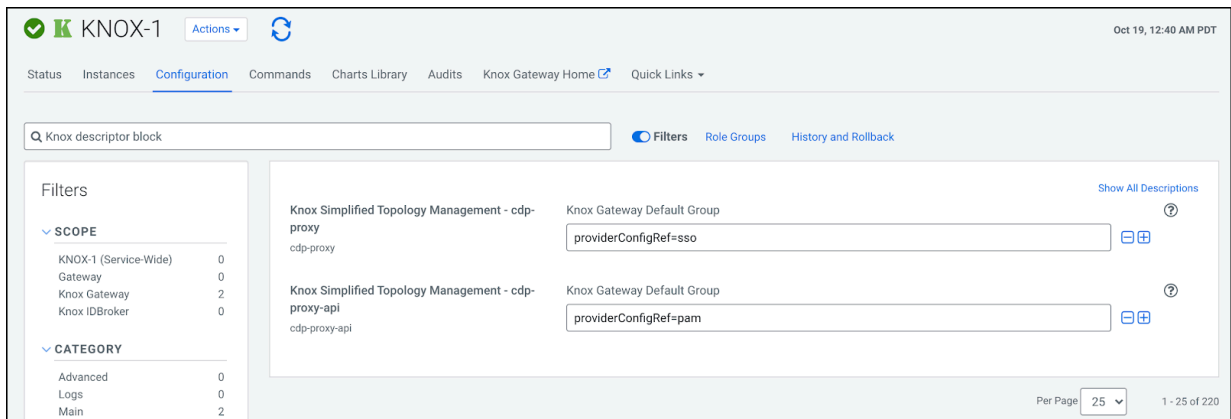
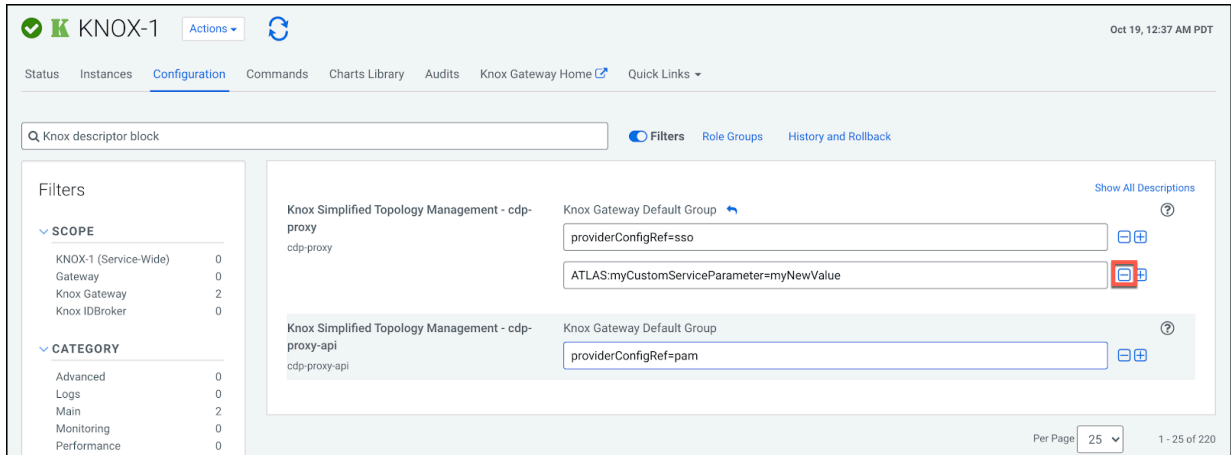


2. Save your changes.
3. The 'Refresh needed' stale configuration indicator appears; click it and wait until the refresh process completes.

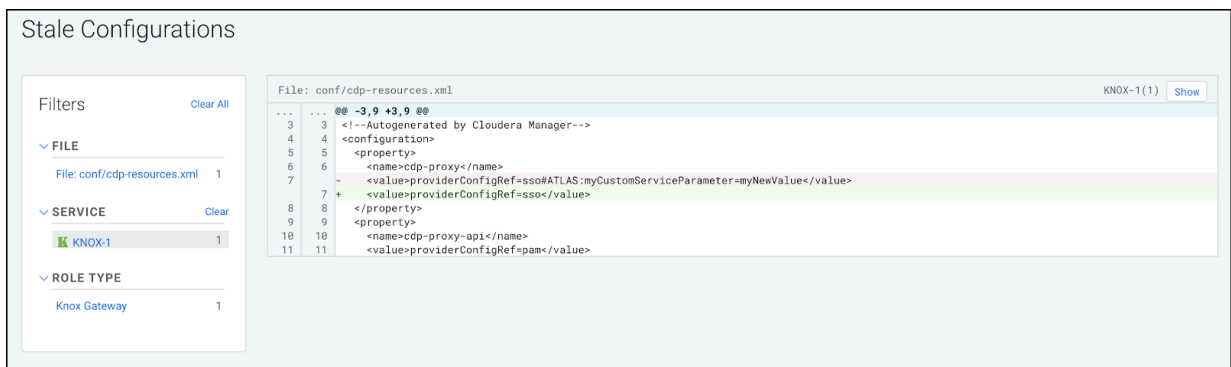


Procedure

- From Cloudera Manager Knox Configuration, remove the service parameter in the Knox Simplified Management - cdp-proxy panel. Click the minus (-) sign next to Atlas:myCustomServiceParameter=myNewValue.



- Save your changes.
- The 'Refresh needed' stale configuration indicator appears; click it and wait until the refresh process completes.



4. Validate that custom service parameter got removed with the changes by navigating to the following URL: `https://$KNOX_GATEWAY_HOST:$PORT/$GATEWAY_PATH/admin/api/v1/topologies/cdp-proxy`.

```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<topology>
  <uri>https://gce.cloudera.com:8443/gateway/cdp-proxy</uri>
  <name>cdp-proxy</name>
  <timestamp>1581950909000</timestamp>
  <generated>true</generated>
  <gateway>...</gateway>
  <service>
    <role>ATLAS</role>
    <url>http://gce.cloudera.com:31000</url>
  </service>
  <service>
    <role>ATLAS-API</role>
    <url>http://gce.cloudera.com:31000</url>
  </service>
</topology>

```

Load balancing for Apache Knox

Knox offers load balancing using a simple round robin algorithm which prevents load on one specific node.

- For services that are stateless, Knox loadbalances them using a simple round robin algorithm which prevents load on one specific node.
- For services that are stateful (i.e., require sessions, such as Ranger and Hive,) sessions are loadbalanced using a round robin algorithm, where each new session will use a different host and all the requests in the same session will be routed to the same host. This will continue until a session terminates or there is a failover.
- In case of failover, services that are stateful will return error response 502.

This behavior is configurable and can be changed by tuning various flags in Knox HA provider for the respective services.

Load balancing vs high availability (HA)

Currently, Knox offers load balancing using a simple round robin algorithm which prevents load on one specific node.

Because we do not support session persistence, this is not true HA, as there could be a case where stateful service will not failover to other node.

Supported services

The following services support Knox load balancing:

- Hive
- Impala
- Oozie
- Phoenix
- Ranger
- Solr

Default enabled values

The following default values are enabled in the Knox topology. API is located in `cdp-proxy-api.xml`; UI is located in `cdp-proxy.xml`.

- Hive
 - API: `enableStickySession=true;noFallback=true;enableLoadBalancing=true`
- Phoenix
 - API: `enableStickySession=true;noFallback=true;enableLoadBalancing=true`
- Ranger
 - API: `enableStickySession=false;noFallback=false;enableLoadBalancing=true`
 - UI: `enableStickySession=true;noFallback=true;enableLoadBalancing=true`
- Solr
 - API: `enableStickySession=false;noFallback=false;enableLoadBalancing=true`
 - UI: `enableStickySession=true;noFallback=true;enableLoadBalancing=true`

Generate and configure a signing keystore for Knox in HA

When Knox is installed on more than one instance (i.e., when Knox is running in HA), then signing keystore configurations must be set in Cloudera Manager.

Procedure

1. Generate your own certificate and keystore file, and then copy to `/var/lib/knox/gateway/data/security/keystores/`.
For more details, see [Manually Configuring TLS Encryption for Cloudera Manager](#).
2. Set the following values:
 - `gateway_signing_keystore_name`: the filename of keystore file that contains the signing keypair.
 - `gateway_signing_keystore_type`: the type of the keystore file where the signing keypair is stored. In non-FIPS environments, this should be PKCS12.
 - `gateway_signing_key_alias`: the alias for the signing keypair within the keystore.
3. If you do not want the master secret to be used, you can set an alias for the password to the keystore file that holds the signing keypair.
 - a) Go to [Saving Aliases](#) and follow the instructions.
 - b) From Cloudera Manager Knox Configuration Knox Service (or Gateway) Advanced Configuration Snippet (Safety Valve) for `conf/gateway-site.xml_service_safety_valve:`, configure `gateway.signing.keystore.password.alias` to the alias previously defined.

Knox Gateway token integration

You can use the Apache Knox homepage to generate and manage Knox Gateway tokens for Cloudera Data Platform.

Related Information

[Knox token management \(in v1.6.0 and above\)](#)

Overview

Instead of using a basic username/password pair, you can improve security by generating Knox Gateway tokens. Tokens are more secure than plaintext username/password because they are signed, anonymized from the source data, and have a specified lifetime (by default, one hour).

About Knox gateway tokens

Before CDP 7.2.14, Knox on CDP Public Cloud had two default topologies: `cdp-proxy` and `cdp-proxy-api`. To enable passcode tokens, a third Knox topology was added: `cdp-proxy-token`. While very similar to `cdp-proxy-api`, the authentication provider for `cdp-proxy-token` is configured with the JWTFederation provider, so that newly generated tokens can be used.

View Knox token integration

Knox token integration can be accessed via Cloudera Manager or the Knox homepage:

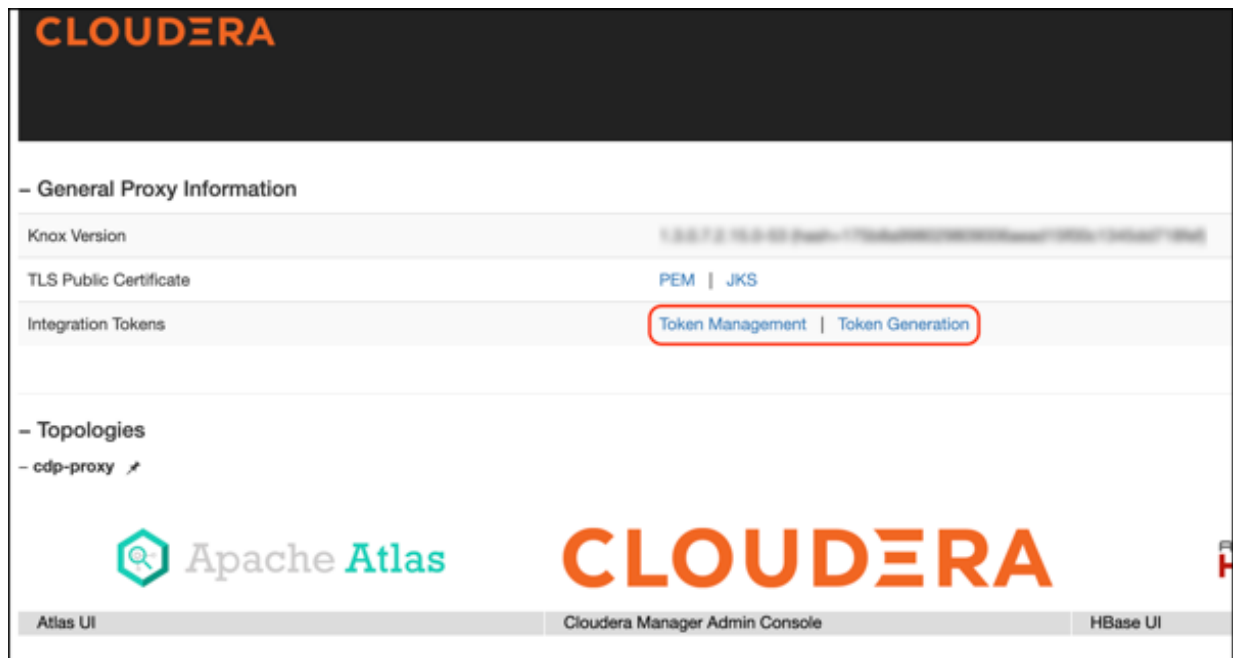
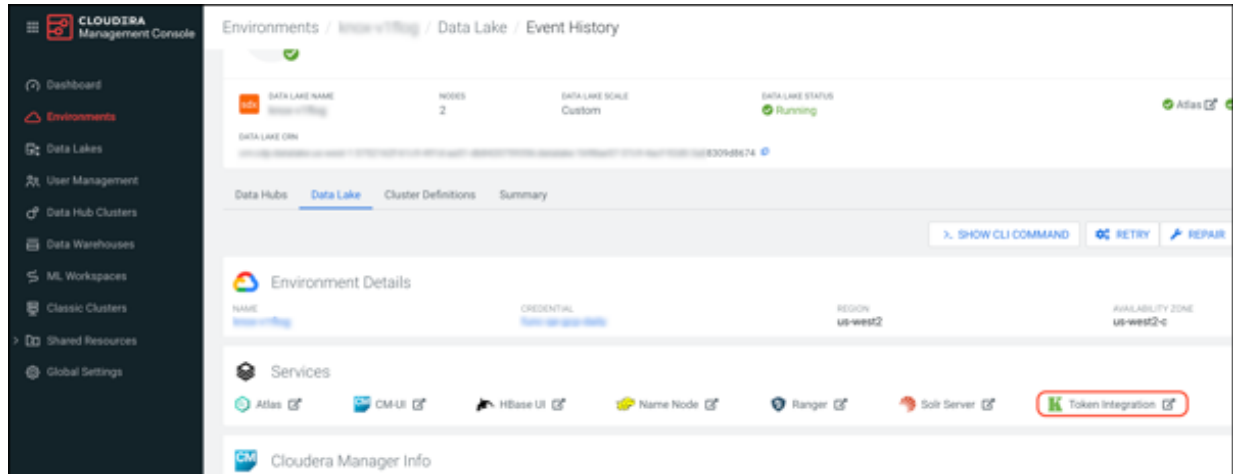
- (Recommended) Cloudera Manager: Cloudera Manager Clusters Knox Configuration and search for “Knox Token Integration”.

The screenshot displays the Cloudera Manager interface for configuring Knox Gateway token integration on cluster KNOX-1. The page is titled "KNOX-1" and shows the "Configuration" tab. A search bar at the top contains "Knox Token Integration". On the left, there is a "Filters" sidebar with sections for SCOPE (KNOX-1, Gateway, Knox Gateway, Knox IDBroker) and CATEGORY (Main, Advanced, Database, Logs, Monitoring, Performance, Ports and Addresses, Resource Management, Security, Stacks Collection). The main content area lists several configuration sections:

- Knox Token Integration - Token State Service Implementation:** Shows the selected implementation as `org.apache.knox.gateway.services.token.impl.JDBCTokenStateService` under the "Knox Gateway Default Group".
- Knox Token Integration - Configured Token TTL:** Shows the TTL set to 1 hour(s).
- Knox Token Integration - Token Type:** Shows the token type set to JWT.
- Knox Token Integration - Allowed Token Management Implementations:** Shows the allowed implementations as `JDBCTokenStateService, AliasBasedTokenStateService`.
- Knox Token Integration - Enable Lifespan Input:** Shows the checkbox for enabling lifespan input fields is unchecked.
- Knox Token Integration - Eviction Grace Period:** Shows the grace period set to 0 day(s).

Each configuration item includes a description and a link to hide all descriptions. The top right corner shows the date and time: Nov 8, 12:31 PM UTC.

- Navigate to the Management Console service > Data Lakes > (Your cluster) > Token Integration (under the Services tab). This will bring you to the Knox homepage. There are two new links on your Knox homepage: Token Management and Token Generation.



Knox token integration in CDP works out of the box using the Knox Token Generation page. However, the token integration API can be re-used in your own custom topology.



Attention: The only restriction of the above approach is that your custom topology must not use the HadoopAuth authentication provider because it won't work with the KNOXTOKEN service due to a known issue (which will be fixed in future releases).

Token configurations

The default configurations for Knox token integration are as follows.

Default configurations

Table 7: Default token configurations

| Property | Sample values | Default |
|--|---|---|
| Token State Service Implementation | Knox's internal implementation of its own token state service. | org.apache.knox.gateway.services.token.impl.JDBCTokenStateService |
| Allowed Token Management Implementations | A list of implementation names that Knox considers allowed on its own token generation page. | JDBCTokenStateService, AliasBasedTokenStateService |
| Configured Token TTL | The value of "knox.token.ttl" in the homepage topology. | 1 hour |
| Token Type | This is an optional configuration parameter to indicate the type of the JWT token that Knox generates. | JWT |
| Enable Lifespan Input | Whether the lifespan input fields are enabled on Knox's token generation page. | false |
| User Limit | The number of tokens a user is allowed to manage. Setting this to -1 indicates unlimited token management. | 10 |
| User Limit Exceeded Action | The action Knox will take if user limit is exceeded while trying to create a new Knox Token. If REMOVE_OLDEST is selected then the oldest token of the user, who the token is being generated for, will be removed. Otherwise, if RETURN_ERROR is selected, Knox will return an error response with 403 error code. | RETURN_ERROR |
| Renewer Whitelist | This is an optional configuration parameter to authorize the comma-separated list of users to invoke the associated token renewal and/or revocation APIs. | empty string |
| JWKS URL | This optional configuration parameter enables end-users to declare their JWKS URL. The JSON Web Key Set (JWKS) is a set of keys containing the public keys used to verify any JSON Web Token (JWT) issued by the authorization server and signed using the RS256 signing algorithm. | empty string |
| Allowed JWS Types | This is an optional configuration parameter to allow a comma-separated list of token types Knox will allow while validating JSON Web Signature (JWS) using JWKS URLs. Defaults to "JWT". The typical customized value is "at +jwt, JWT". | JWT |
| Expected Principal Claim | If that configuration parameter is defined, Knox will use this to get the value of this claim from the submitted JWT upon verification instead of using the default principal. | empty string |
| Expected JWT Signature Algorithm | Indicates the expected signature algorithm Knox should use to verify the submitted JWT's signature. If not defined, Knox will use 'RS256'. | empty string |
| Expected JWT Issuer | Indicates the expected issuer of a received token must match. If not defined, Knox will use 'KNOXSSO'. | empty string |
| Enable Impersonation | Indicates if Knox Token impersonation is enabled. | false |

| Property | Sample values | Default |
|-----------------|---|--------------|
| Proxyuser Block | <p>If Knox token impersonation is enabled, it allows the specified user(s) to impersonate members of certain groups from specific hosts. For example, the following values allow a user named "changeme" to impersonate members of any group from any host:</p> <ul style="list-style-type: none"> "knox.token.proxyuser.changeme.hosts": "*" "knox.token.proxyuser.changeme.groups": "*" <p>The values for this property must be in JSON key/value format.</p> | empty string |

Default configurations seen from Cloudera Manager:

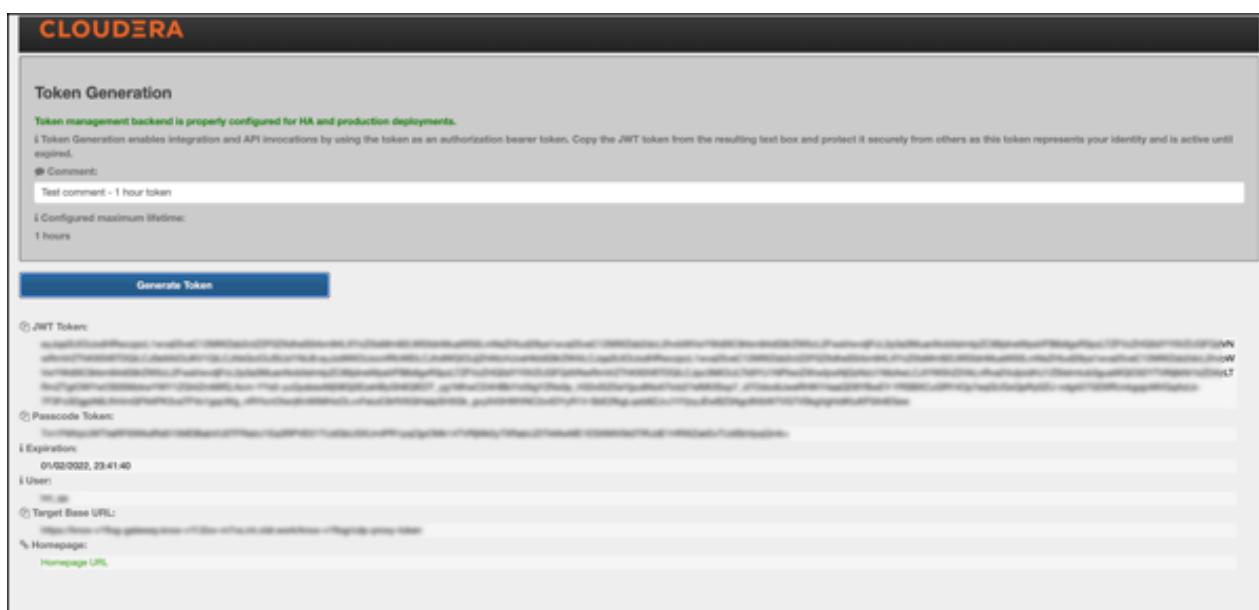
The screenshot shows the Cloudera Manager interface for configuring Knox Token Integration. The search bar contains 'Knox Token Integration'. The left sidebar shows filters for SCOPE (KNOX-1, Gateway, Knox IDBroker) and STATUS (Error, Warning, Edited, Non-Default, Include Overrides). The main configuration area lists several properties:

- Knox Token Integration - Token State Service Implementation:** Set to `org.apache.knox.gateway.services.token.impl.JDBCTokenStateService`.
- Knox Token Integration - Configured Token TTL:** Set to `1` hour(s).
- Knox Token Integration - Token Type:** Set to `JWT`.
- Knox Token Integration - Allowed Token Management Implementations:** Set to `JDBCTokenStateService, AliasBasedTokenStateService`.
- Knox Token Integration - Enable Lifespan Input:** Checked.
- Knox Token Integration - Eviction Grace Period:** Set to `day(s)`.

| | | |
|---|---|--|
| <p>Knox Token Integration - User Limit gateway.knox.token.limit.per.user gateway_knox_token_limit_per_user</p> | <p>Knox Gateway Default Group</p> <input type="text" value="10"/> | <p>The number of tokens a user is allowed to manage. Setting this to -1 indicates unlimited token management.</p> |
| <p>Knox Token Integration - User Limit Exceeded Action gateway.knox.token.user.limit.exceeded.action gateway_knox_token_user_limit_exceeded_action</p> | <p>Knox Gateway Default Group</p> <p><input type="radio"/> REMOVE_OLDEST</p> <p><input checked="" type="radio"/> RETURN_ERROR</p> | <p>The action Knox will take if user limit is exceeded while trying to create a new Knox Token. If REMOVE_OLDEST is selected then the oldest token of the user, who the token is being generated for, will be removed. Otherwise, if RETURN_ERROR is selected, Knox will return an error response with 403 error code.</p> |
| <p>Knox Token Integration - Renewer Whitelist gateway.knox.token.renewer.whitelist gateway_knox_token_renewer_whitelist</p> | <p>Knox Gateway Default Group</p> <input type="text"/> | <p>This is an optional configuration parameter to authorize the comma-separated list of users to invoke the associated token renewal and/or revocation APIs.</p> |
| <p>Knox Token Integration - JWKS URL gateway.knox.token.jwks.url gateway_knox_token_jwks_url</p> | <p>Knox Gateway Default Group</p> <input type="text"/> | <p>This optional configuration parameter enables end-users to declare their of JWKS URL. The JSON Web Key Set (JWKS) is a set of keys containing the public keys used to verify any JSON Web Token (JWT) issued by the authorization server and signed using the RS256 signing algorithm.</p> |
| <p>Knox Token Integration - Allowed JWS Types gateway.knox.token.allowed.jws.types gateway_knox_token_allowed_jws_types</p> | <p>Knox Gateway Default Group</p> <input type="text" value="JWT"/> | <p>This is an optional configuration parameter to allow a comma-separated list of token types Knox will allow while validating JSON Web Signature (JWS) using JWKS URLs. Defaults to 'JWT'. Typical customized value is 'at+jwt, ...'</p> |
| <p>Knox Token Integration - Expected Principal Claim gateway.knox.token.expected.principal.claim gateway_knox_token_expected_principal_claim</p> | <p>Knox Gateway Default Group</p> <input type="text"/> | <p>If that configuration parameter is defined, Knox will use this to get the value of this claim from the submitted JWT upon verification instead of using the default principal.</p> |
| <p>Knox Token Integration - Expected JWT Signature Algorithm gateway.knox.token.expected.jwt.signature.algorithm gateway_knox_token_expected_jwt_signature_algorithm</p> | <p>Knox Gateway Default Group</p> <input type="text"/> | <p>Indicates the expected signature algorithm Knox should use to verify the submitted JWT's signature. If not defined, Knox will use 'RS256'.</p> |
| <p>Knox Token Integration - Expected JWT Issuer gateway.knox.token.expected.jwt.issuer gateway_knox_token_expected_jwt_issuer</p> | <p>Knox Gateway Default Group</p> <input type="text"/> | <p>Indicates the expected issuer of a received token must match. If not defined, Knox will use 'KNOXSSO'.</p> |
| <p>Knox Token Integration - Enable Impersonation gateway.token.generation.enable.impersonation gateway_token_generation_enable_impersonation</p> | <p><input checked="" type="checkbox"/> Knox Gateway Default Group ↗</p> | <p>Indicates if Knox Token impersonation is enabled.</p> |
| <p>Knox Token Integration - Proxyuser Block gateway.knox.token.impersonation.proxyuser.block gateway_knox_token_impersonation_proxyuser_block</p> | <p>Knox Gateway Default Group</p> <input type="text" value='"knox.token.proxyuser.changeme.hosts": "*"'/> | <p>Proxyuser configuration used in Knox's 'homepage' topology for token impersonation purposes. Must conform a valid JSON key-value format!</p> |
| | <input type="text" value='"knox.token.proxyuser.changeme.groups": "*"'/> | |

Rows per page: 25 1 - 16 of 16

Default configurations seen from the Knox homepage UI:



Database connection properties

Optional database connection properties that you can declare individually:

- gateway.database.type: Set to postgresql or mysql.
- gateway.database.host: Host where your DB server is running.
- gateway.database.port: Port that your DB server is listening on.
- gateway.database.name: Name of the database you are connecting to.

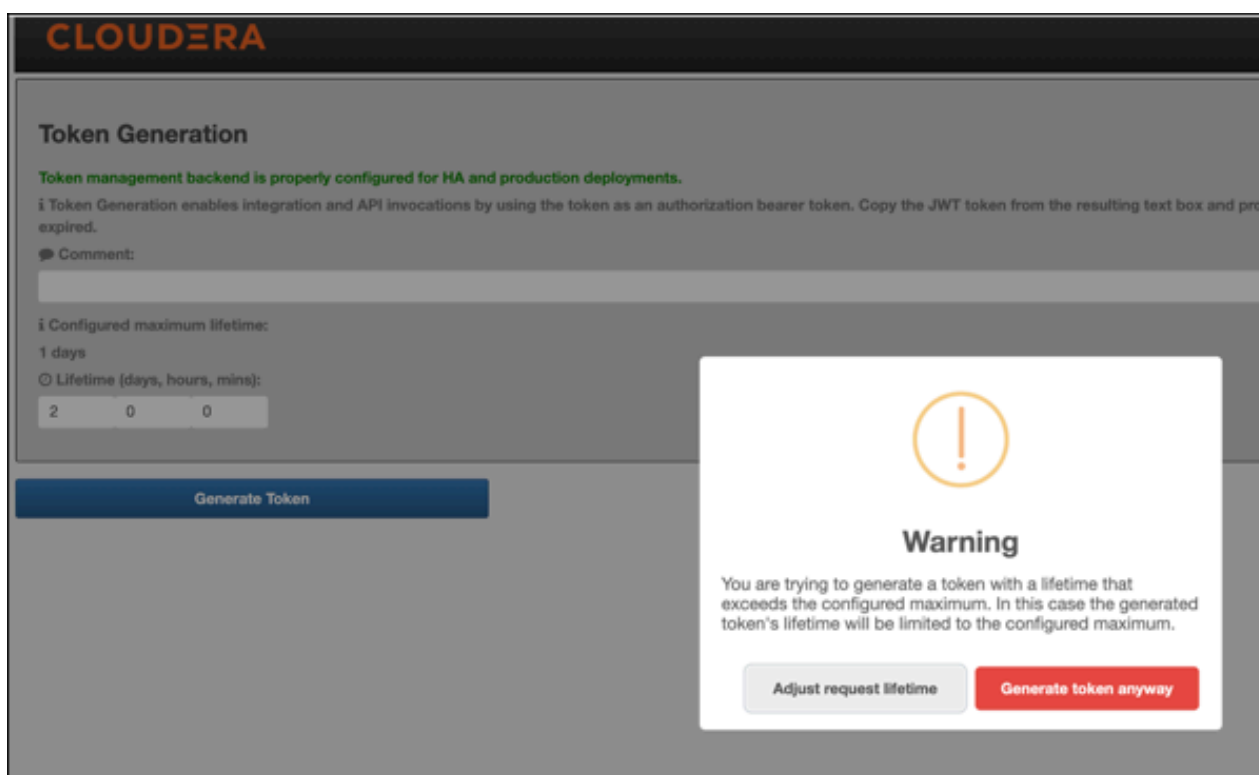


Important: From CDP Private Cloud Base 7.1.9 SP1 CHF1 release onwards, if you are using a MySQL database, then the JDBC driver must be installed manually to /usr/share/java. For more information, see [Installing the MySQL JDBC Driver](#).

Token TTL details

Out of the box, Knox will display the custom lifetime spinners on the Token Generation page. However, they can be hidden by disabling the Knox Token Integration - Enable Lifespan Input checkbox on the CM UI. Given that input property, and the configured maximum lifetime property, the generated token can have the following TTL value:

- If there is no configured token TTL and lifespan inputs are disabled, the default TTL is used (30 seconds).
- If there is configured TTL and lifespan inputs are disabled, the configured TTL is used.
- If there is configured TTL and lifespan inputs are enabled and lifespan inputs result in a value that is less than or equal to the configured TTL, the lifespan query param is used.
- If there is configured TTL and lifespan inputs are enabled and lifespan inputs result in a value that is greater than the configured TTL, the configured TTL is used.



Generate-jwk options

CM automatically creates a token hash key for you. But if you want to do this manually, such as when scripting, configure the `knox.token.hash.key` alias with:

```
generate-jwk --saveAlias knox.token.hash.key
```

This generates a JSON Web Key using the supplied algorithm name.

Table 8: Options

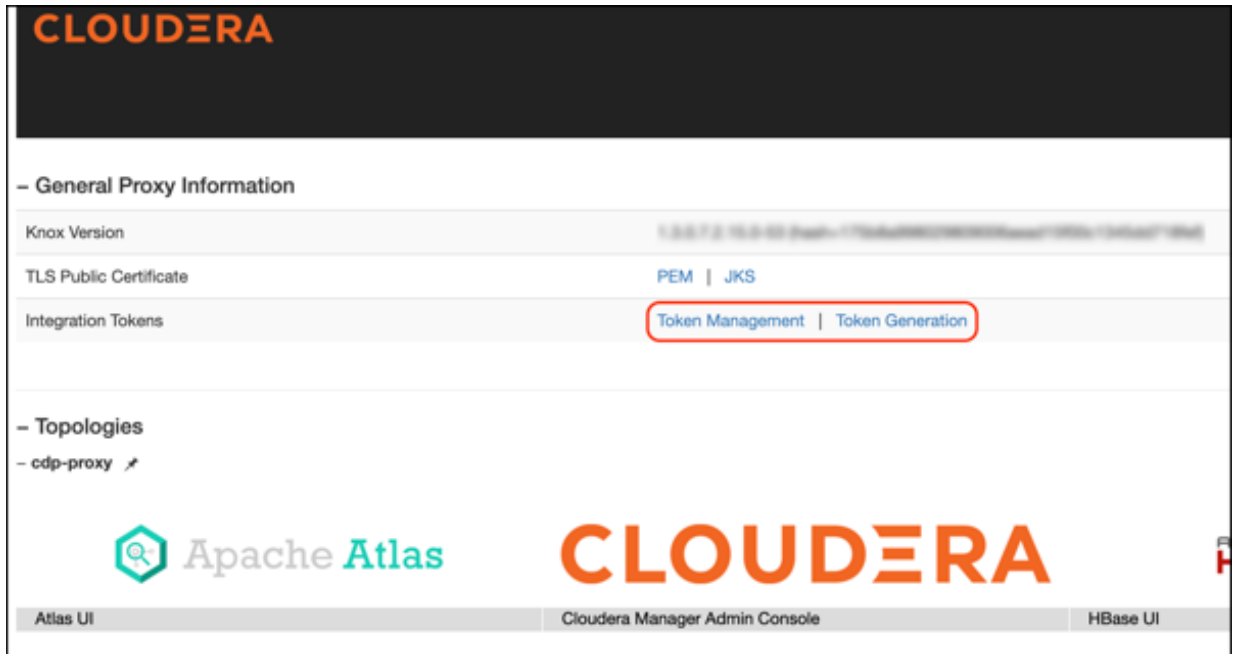
| Option | Description | Sample values |
|------------------------|--|--|
| <code>jwkAlg</code> | (Optional) The desired JSON Web Signature algorithm name. Determines if the gateway-level alias is configured with a 256, 384, or 512-bit length JWK. | HS256 (Default) HS384 HS512 |
| <code>saveAlias</code> | (Optional, Recommended) Given alias name used to save the generated JWK, instead of printing this sensitive information on the screen. | <code>knox.token.hash.key</code> |
| <code>topology</code> | (Optional) Name of the topology (i.e., the cluster) to be used when saving the JWK as an alias. If none specified, the alias is going to be saved for the Gateway. | <code>cdp-proxy</code> (Default) <code>cdp-proxy api</code> |

Generate tokens

How to generate Knox gateway tokens from the Knox homepage.

Procedure

1. To access Knox generation management, go to `https://KNOX_GATEWAY_HOST:PORT/GATEWAY_PATH/homepage/home`, e.g. `https://localhost:8443/gateway/homepage/home`. Click on Token Generation.



2. The following sections are displayed on the page:

- Status bar: Message about the configured token state backend. There are 3 different statuses:
 - ERROR: Displayed in red. Indicates a problem with the service backend which makes the feature not work. Usually, this is visible when end-users configure JDBC token state service, but they make a mistake in their DB settings.
 - WARN: Displayed in yellow. Indicates that the feature is enabled and working, but there are some limitations.
 - INFO: Displayed in green. Indicates when the token management backend is properly configured for HA and production deployments.
- Information label: Explains the purpose of the **Token Generation** page.
- Comment: Optional input field that allows end-users to add meaningful comments (mnemonics) to their generated tokens. The maximum length is 255 characters.
- Configured maximum lifetime: Informs the clients about the `knox.token.ttl` property set in the homepage topology (defaults to 1 day(s)). If that property is not set (e.g. someone removes it from the homepage topology), Knox uses a hard-coded value of 30 seconds (aka. default Knox token TTL).
- Custom maximum (token) lifetime: Can be set by adjusting the days/hours/minutes fields. The default configuration will yield one hour.

The screenshot shows the Cloudera Token Generation page. At the top, the Cloudera logo is visible. Below it, the page title is "Token Generation". A green message states: "Token management backend is properly configured for HA and production deployments." Below this, a small informational text explains that the token generation enables integration and API invocations by using the token as an authorization bearer token. There is a "Comment:" label followed by a text input field. Below that, the "Configured maximum lifetime:" is shown as "1 days". A "Lifetime (days, hours, mins):" section contains three input fields with values "0", "1", and "0". At the bottom, there is a blue "Generate Token" button.

If Knox Token Integration - Enable Impersonation is set to true, another input field is shown on the UI called Generating token for (impersonation).

Using that input field our customers should be able to generate tokens on behalf of other users. For this to work, the Knox Token Integration - Proxyuser Block property has to be configured properly.

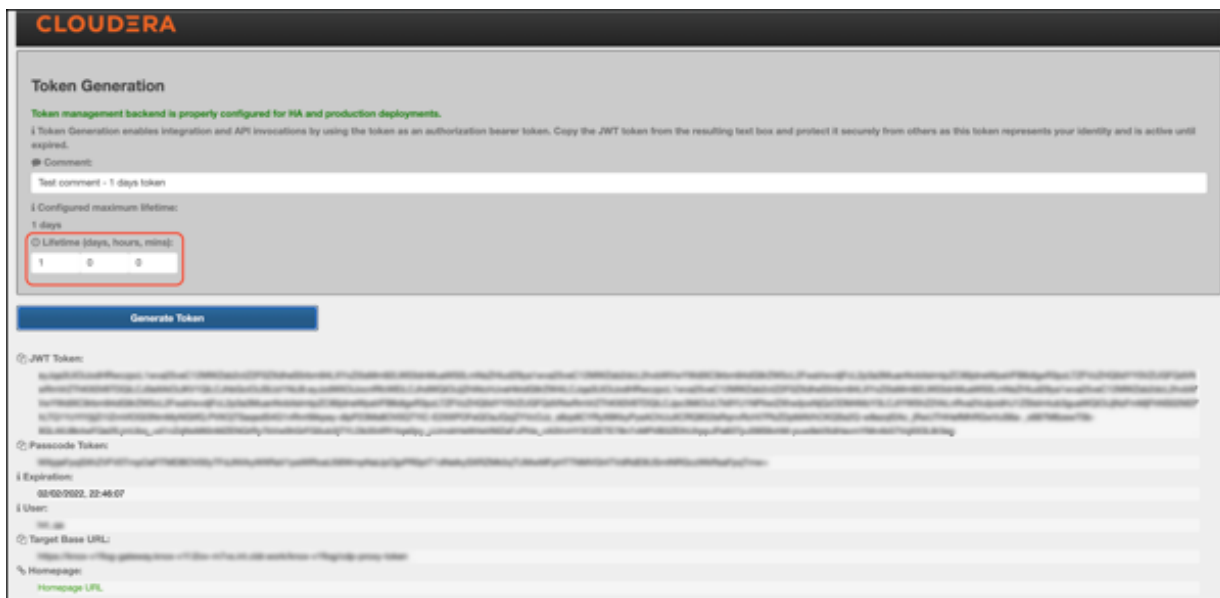


Important: If Knox is behind a Load Balancer and Token Impersonation support is used while generating tokens (that input field is populated with a username), the Load Balancer host must be added to the Proxy User configuration too. If the user wants to decline requests from a specific host, then that can be configured on the Load Balancer side.

This screenshot is similar to the previous one but includes an additional input field. Below the "Lifetime" section, there is a label "Generating token for (impersonation):" followed by a text input field. The "Generate Token" button remains at the bottom.

For more information, see [Knox Apache User-guide: Token impersonation](#)

3. Click Generate Token.



4. Use the token to authenticate your request. Click the icon beside your choice on the page to copy the value to the clipboard:

- **JWT token:** serialized JWT, fully compatible with the old-style bearer authorization method. You can use it as the 'Token' user:

```
$ curl -ku Token:eyJqa3U[... ]uT5AxQGyMMP3VLGw https://localhost:8443/gate
way/cdp-proxy-token/webhdfs/v1?op=LISTSTATUS

{"FileStatuses":{"FileStatus":[{"accessTime":0,"blockSize":0,"childre
nNum":1,"fileId":16386,"group":"supergroup",
"length":0,"modificationTime":1621238405734,"owner":"hdfs","pathSuffix
":"tmp","permission":"1777","replication":0,
"storagePolicy":0,"type":"DIRECTORY"},{"accessTime":0,"blockSize":0,"chi
ldrenNum":1,"fileId":16387,"group":"supergroup",
"length":0,"modificationTime":1621238326078,"owner":"hdfs","pathSuffix
":"user","permission":"755","replication":0,
"storagePolicy":0,"type":"DIRECTORY"}]}}
```

- **Passcode token:** Serialized passcode token, which can be used as the 'Passcode' user:

```
$ curl -ku Passcode:WkRFMk1XTmh[... ]RVNFpXRTA= https://localhost:8443/ga
teway/cdp-proxy-token/webhdfs/v1?op=LISTSTATUS

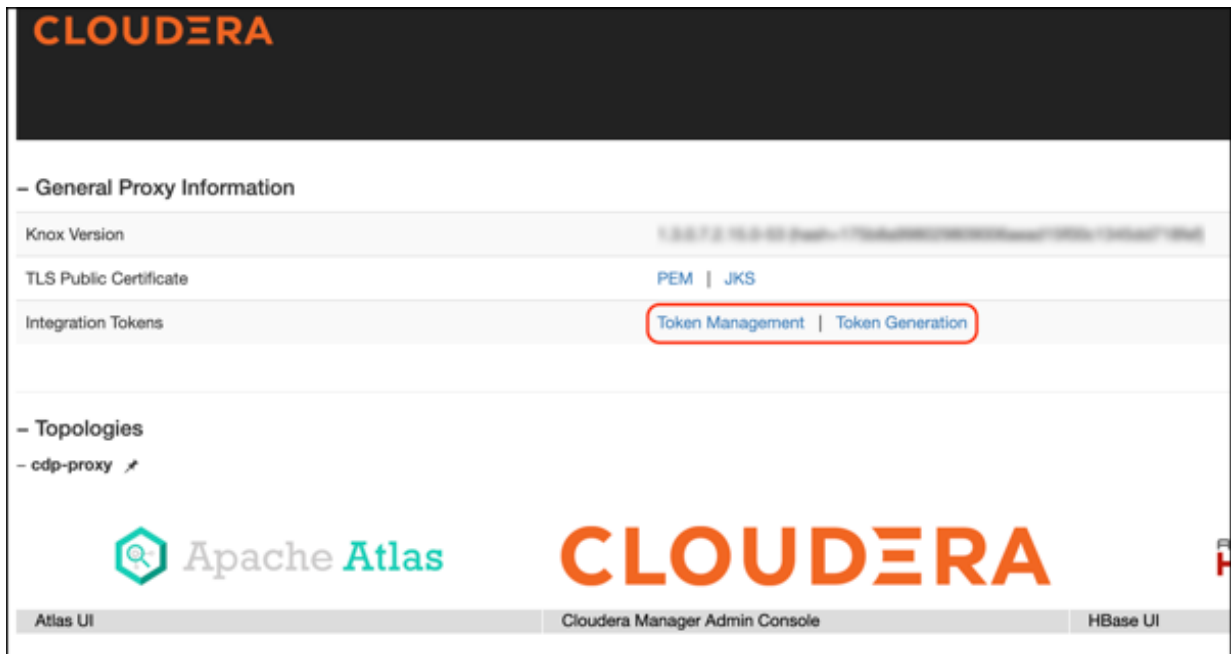
{"FileStatuses":{"FileStatus":[{"accessTime":0,"blockSize":0,"childrenN
um":1,"fileId":16386,"group":"supergroup",
"length":0,"modificationTime":1621238405734,"owner":"hdfs","pathSuffi
x":"tmp","permission":"1777","replication":0,
"storagePolicy":0,"type":"DIRECTORY"},{"accessTime":0,"blockSize":0,"c
hildrenNum":1,"fileId":16387,"group":"supergroup",
"length":0,"modificationTime":1621238326078,"owner":"hdfs","pathSuffi
x":"user","permission":"755","replication":0,
"storagePolicy":0,"type":"DIRECTORY"}]}}
```

Manage Knox Gateway tokens

You can enable, disable, or revoke tokens via the Knox homepage.

Procedure

- To access Knox token management, go to `https://KNOX_GATEWAY_HOST:PORT/GATEWAY_PATH/homepage/home`, e.g. `https://localhost:8443/gateway/homepage/home`. Click on Token Management.



A compact view of all tokens generated within the system is shown in a single table with the following information.

| <input type="checkbox"/> | Token ID | Issued | Expires | User Name | Impersonated | KnoxSSO | Comment | Additional Metadata | Actions |
|-------------------------------------|--------------------------------------|----------------------|----------------------|-----------|-------------------------------------|-------------------------------------|-----------------|---------------------|------------------|
| <input type="checkbox"/> | b67434df-7972-413f-924f-18735dc26263 | 07/11/2023, 13:18:44 | 08/11/2023, 13:18:44 | knoxui | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | | | Disable |
| <input type="checkbox"/> | fb905e3-1f2c-4205-8505-d3ddc5ca0f6 | 07/11/2023, 13:35:02 | 07/11/2023, 14:35:02 | knoxui | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | token 1 hour | | Enable Revoke |
| <input checked="" type="checkbox"/> | 6d279f69-7404-4dbf-bcd1-df73dc3f51d5 | 07/11/2023, 13:35:22 | 07/11/2023, 13:38:22 | knoxui | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | token 3 minutes | | Revoke |
| <input checked="" type="checkbox"/> | 66a9b7ba-7bac-4f8c-b0a5-89a995ce95f8 | 07/11/2023, 13:37:54 | 07/11/2023, 14:37:54 | bob | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | token for bob | | Disable Revoke |
| <input checked="" type="checkbox"/> | d9fd1384-491f-4425-873a-fae1497ad3f | 07/11/2023, 13:39:16 | 07/11/2023, 14:39:16 | knoxui | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | token 1 hour | | Disable Revoke |

Items per page: 25 1 - 5 of 5

Revoke Selected Tokens

⚠ Expired tokens cannot be disabled in batches (nor individually).


⚠ Expired tokens cannot be enabled in batches (nor individually).

- Each row starts with a selection checkbox for batch operations (except for disabled KnoxSSO cookies, as there is no point in doing anything with them).
- A unique token identifier. Disabled token's Token ID value is shown in orange.
- Information on when the token was created and when it will expire.
 - If the token is already expired, the expiration time is shown in red.
 - If the token is still valid, the expiration time is shown in green.
- Username indicates the user for whom the token is created for.
- Impersonated is a boolean flag indicating if this is an impersonated token:
 - Green check: Yes, this is impersonated. You'll see the user who created the token under the icon.
 - Red cross: No, this is not an impersonated token.

- f. KnoxSSO is another boolean flag that indicates if this token is created by the KNOXSSO service if the feature was enabled. The line is shown in bold if the token in this line represents the SSO cookie currently used as the authentication token to login to the Token Management page.
 - green check: yes, this is KnoxSSO cookie (token)
 - red cross: no, this is not a KnoxSSO cookie (it was created by a regular token API call or on the Token Generation page or the feature is disabled)
- g. Comment: users may add a short comment to the tokens they create to make it easier for them to distinguish certain tokens later (e.g. "1-hour token for user XY")
- h. Additional Metadata : In some cases, it's beneficial to add different metadata to the generated token as a key-value pair (e.g. shouldBeRemovedBy=09_Nov_2023). One token can have more than one associated metadata. In this column, we display that information.
- i. In the Actions column, you will see
 - The enable/disable/revoke actions are visible for impersonated tokens too
 - KnoxSSO cookies cannot be revoked nor re-enabled.

In order to refresh the table, you can use the Refresh icon above the table (if you generated tokens on another tab for instance).

2. You can perform batch operations on the tokens. When at least one token is selected, the following buttons are shown under the table:
 - Disable - when executed, all the selected tokens become disabled (if they were disabled originally, they will remain disabled). Please note this option is shown only, if there is no expired token selected (i.e. batch disablement only works with live tokens).
 - Enable - when executed, all the selected tokens become enabled (if they were enabled originally, they will remain enabled). Please note this option is shown only, if there is no expired token selected (i.e. batch enablement only works with live tokens).
 - Revoke - when executed, all the selected tokens will be revoked. Please note this option is shown only, if there is no KnoxSSO cookie (token) selected (i.e. batch revocation only works with regular tokens).

 **Note:** If the selected tokens contain any that cannot be disabled/enabled (expired tokens) or revoked (Knox SSO Cookies), an informational message is displayed below the batch operation buttons indicating the root cause.

3. You can use the Search by field to narrow down tokens by :
 - Token ID
 - User Name (either own user name or impersonated)
 - Comment
 - Additional Metadata
4. You can view the disabled Knox cookies or only your tokens by using the following toggle buttons.
 - Show Disabled KnoxSSO Cookies - This is true by default. Since disabled KnoxSSO cookies remain in the underlying token state service until they expire, it may bother users to see them in the tokens table. Flipping this toggle button helps to hide them.
 - Show My Tokens Only - this toggle button is only visible to users, who can see all tokens. By default, this is false. Enabling it will filter the tokens table in a way such that it will contain tokens only that were generated for the logged-in user (impersonated or not).
5. Click the Refresh icon above the table.

Knox Token API

The Knox Token Service enables the ability for clients to acquire the same JSON Web Token (JWT) that is used for KnoxSSO with WebSSO flows for UIs to be used for accessing REST APIs.

Introduction

By acquiring the token and setting it as a Bearer token on a request, a client is able to access REST APIs that are protected with the JWTProvider federation provider.

In CDP, the only Knox topology that is properly configured for Knox Token management is the homepage topology, so that the following samples will assume KnoxSSO authentication. To use the Knox Token API, you should copy the `hadoop-jwt` cookie from your browser and export it as an ENV variable :

```
export HJWT="eyJraWQiOiJtSWdPbjRPSWZlUmZlRlVLWjVSB3dxbFh3SGUycGUcm9lWj1EX1BLUUNiIiwiaXNzIjoieS05PWFNTTyIsImV4cCI6MTcwMjU1MjE1OCwibWFuYXd1ZC50b2t1biI6ImZhbHN1Iiwia25veC5pZCI6IjY5M2EwMmE3LTVhYTgtNDA2MS1iYjMyLTA4MDk4Yzd1MTkxYiJ9.tLzmxSd64bAQGpKdETsNXaaDBUKyMZzp0j0YNi-14Jmcm1oG5PerUt00EmLWQnsDBuqtzExkR-g8metxwyIwjV61rqZRXLfycrN8x-nMTCEdxjcjtMegIS3XyETut8MRx8nk6WPVcBlwGHnOCG52CvxsVBe7pUFD4jYybGzF_WlkPDzPjSRcdQ3xRFDq2IFt7RxOIye_50ZdMLbZBm9rNi0RErgdrLKJse68fly-58BcfquubFgWUA0Z0QND7Gg3lPBzyBOhe_5YA23jQsicgvctc-HhNkY6W2RP-qpXmgjInGcy7dnpvbHXQNfA8cXffDQA6e3bFrTHpJNHgpEsGeg"
```

Alternatively, you can create a custom topology with another authentication provider (PAM, for instance) and use that topology instead of homepage in the following examples.



Important: KnoxToken API 'v1' is no longer supported in 7.2.18 and subsequent CDP releases. All the samples use the new 'v2' endpoints with the corresponding HTTP methods (GET, PUT, DELETE).

Acquire a token

```
$ curl -ik --cookie "hadoop-jwt=$HJWT" -X GET https://[***HOST_NAME***]:8443/gateway/homepage/knoxtoken/api/v2/token
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 11:15:16 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 3073{"access_token":"eyJqa3UiOiJo...4BnSw","token_id":"8f7de7f7-a094-4ba4-b64d-f21be86f10eb","managed":"true","target_url":"cdp-proxy-token","homepage_url":"homepage/home?profile=token&topologies=cdp-proxy-token","endpoint_public_cert":"MIIElJCCAz...cXff0","token_type":"Bearer","expires_in":1702469717045,"passcode":"T0dZM1pH...U1ESXo="}
```

In the result JSON, end-users can find the following information:

- `accessToken`: this is the serialized JWT and is fully compatible with the old-style Bearer authorization method. End-users might want to use it as the 'Token' user in the `cdp-proxy-token` topology for authentication purposes.
- `passcode`: this is another sensitive data, the serialized passcode token, which end-users can use as the 'Passcode' user for authentication purposes
- `token_id`: the unique identifier of the token within Knox
- `managed`: this is a boolean flag indicating if the token is managed. Managed tokens can be renewed, revoked, disabled, and enabled. By default, in CDP, the homepage topology is configured to manage tokens.
- `expires`: indicates the expiration time of this token. This may be updated with token renewal.

Renew a token

Currently, renewing a token is feasible only if you pass the value of the `access_token` field as the request payload. For instance:

```
$ export KNOX_TOKEN="eyJqa3UiOiJo...4BnSw"
```

```
$ curl -ik --cookie "hadoop-jwt=$HJWT" -H "X-XSRF-HEADER:valid" -d $KNOX_TOKEN -X PUT 'https://[***HOST_NAME***]:8443/gateway/homepage/knoxtoken/api/v2/token/renew'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 15:22:01 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: text/plain
Content-Length: 54

{
  "renewed": "true",
  "expires": "1702567321838"
}
```

The result JSON will tell you if the renewal was a success (this information is stored in the renewed field) and the (new) expiration time. In case of an invalid/unknown token, you should expect a response like this :

```
{
  "renewed": "false",
  "error": "Unknown token: 9caf743e-1e0d-4708-a9ac-a684a576067c"
}
```



Note: Token Renewal is allowed only for a certain set of users, which end-users can define using the Knox Token Integration - Renewer Whitelist Knox configuration on the CM UI.

The screenshot shows the Cloudera Management UI for the 'Knox Token Integration - Renewer Whitelist' configuration. The page has a search bar at the top with the text 'Knox Token Integration - Renewer Whitelist'. To the right of the search bar are links for 'Filters', 'Role Groups', and 'History & Rollback'. Below the search bar is a 'Filters' section with a 'SCOPE' dropdown menu. The 'SCOPE' menu is open, showing a list of scopes: 'KNOX-1 (Service-Wide)' with a count of 0, 'Gateway' with a count of 0, 'Knox Gateway' with a count of 1, and 'Knox IDBroker' with a count of 0. The main configuration area shows the 'Knox Token Integration - Renewer Whitelist' configuration. It includes a 'Knox Gateway Default Group' dropdown menu with 'Undo' and 'Show All Descriptions' options. Below this is a text input field containing 'gateway_knox_token_renewer_whitelist' and a 'gateway_knox_token_renewer_whitelist' link. The page number '1 - 1 of 1' is visible at the bottom right.

If the requesting user is an unauthorized caller, you should expect a response like this:

```
{
  "renewed": "false",
  "error": "Caller (myTestUser) not authorized to renew tokens."
}
```

Token revocation

End-users can revoke a token using either the token_id or the access_token fields from the above acquired token response. For instance, the following sample uses the token_id :

```
curl -ik --cookie "hadoop-jwt=$HJWT" -H "X-XSRF-HEADER:valid" -d 'cb538d38-3076-4a7a-90a5-0f26bfff2939a' -X DELETE 'https://[***HOST_NAME***]:8443/gateway/homepage/knoxtoken/api/v2/token/revoke'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 15:31:31 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 24

{
  "revoked": "true"
}
```

```
}

```

In case of an invalid or unknown token, you should expect a similar error like this:

```
{
  "revoked": "false",
  "error": "Unknown token: cb538d38...0f26bff2939a",
  "code": 50
}
```

Token revocation also requires authorization. The same Cloudera Manager configuration should be used that is listed above for token renewals. There is an exception though with revocation: end-users can revoke tokens that belong to them. That is, you can revoke your very own token even if your user name is not defined in the Knox Token Integration - Renewer Whitelist configuration.

Enable/Disable a Token

End-users might need to temporarily disable a token for security purposes and then re-enable the same token. You can do that with the following API calls which use the `token_id` field from the acquired token response:

```
$ curl -ik --cookie "hadoop-jwt=$HJWT" -H "X-XSRF-HEADER:valid" -d '1cad8a13-08e2-4b8a-8076-082678bb641b' -X PUT 'https://[***HOST_NAME***]:8443/gateway/homepage/knoxtoken/api/v2/token/disable'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 15:42:57 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 55
```

```
{
  "setEnabledFlag": "true",
  "isEnabled": "false"
}
```

```
$ curl -ik --cookie "hadoop-jwt=$HJWT" -H "X-XSRF-HEADER:valid" -d '1cad8a13-08e2-4b8a-8076-082678bb641b' -X PUT 'https://[***HOST_NAME***]:8443/gateway/homepage/knoxtoken/api/v2/token/disable'
HTTP/1.1 400 Bad Request
Date: Wed, 13 Dec 2023 15:43:21 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 86
```

```
{
  "setEnabledFlag": "false",
  "error": "Token is already disabled",
  "code": 60
}
```

```
$ curl -ik --cookie "hadoop-jwt=$HJWT" -H "X-XSRF-HEADER:valid" -d '1cad8a13-08e2-4b8a-8076-082678bb641b' -X PUT 'https://[***HOST_NAME***]:8443/gateway/homepage/knoxtoken/api/v2/token/enable'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 15:43:45 GMT
X-Frame-Options: DENY
```

```
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 54
```

```
{
  "setEnabledFlag": "true",
  "isEnabled": "true"
}
```

Fetching user tokens

The KnoxToken API provides a powerful way to fetch/filter previously created tokens. See the following samples :

```
$ curl -ik --cookie "hadoop-jwt=$HJWT" -X GET 'https://[***HOST_NAME***]:8443/gateway/homepage/knoxtoken/api/v2/token/getUserTokens?userName=knoxui'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 15:46:50 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 413
```

```
{ "tokens": [ { "tokenId": "1cad8a13-08e2-4b8a-8076-082678bb641b", "issueTime": "2023-12-13T15:42:17.965+0000", "expiration": "2023-12-13T16:42:17.957+0000", "maxLifetime": "2023-12-20T15:42:17.965+0000", "metadata": { "knoxSsoCookie": false, "customMetadataMap": {}, "createdBy": null, "comment": null, "enabled": true, "userName": "knoxui" }, "maxLifetimeLong": 1703086937965, "issueTimeLong": 1702482137965, "expirationLong": 1702485737957 } ] }
```

Manage Knox metadata

This document describes how to manage Token Metadata.

As indicated in the previous sections, the KNOXTOKEN service maintains some hard-coded token metadata out-of-the-box:

- userName
- comment
- enabled
- passcode
- createdBy (in case of impersonated tokens)

In Cloudera Runtime version 7.2.16, Cloudera has introduced support for a new feature that allows end-users to add accept query parameters starting with the md_ prefix and treat them as Knox Token Metadata.

Example

```
curl -iku admin:admin-password -X GET 'https://$KNOX_GATEWAY_HOST:8443/gateway/sandbox/knoxtoken/api/v1/token?md_notebookName=accountantKnoxToken&md_shouldBeRemovedBy=31March2022&md_otherMeaningfulMetadata=KnoxIsCool'
```

When such a token is created by Knox, the following metadata should be saved:

- notebookName=accountantKnoxToken
- shouldBeRemovedBy=31March2022
- otherMeaningfulMetadata=KnoxIsCool

It will not only enable Knox to save these metadata, but will also enable Knox's existing getUserTokens API endpoint to fetch basic token information using the supplied metadata name besides the username information.



Note: The getUserTokens API returns tokens if any of the supplied metadata exists for the given token. Metadata values may or may not be matched: you can either use the * wildcard to match all metadata values with a given name or you can further filter the stored metadata information by specifying the desired value.

Example:

```
curl -iku admin:admin-password -X GET 'https://$KNOX_GATEWAY_HOST:8443/gateway/sandbox/knoxtoken/api/v1/token/getUserTokens?userName=admin&md_notebookName=accountantKnoxToken&md_name=*'
```

It will return all Knox tokens where metadata with notebookName exists and equals accountantKnoxToken OR metadata with name exists.

Another Sample:

1. Create token1 with md_Name=reina&md_Score=50
2. Create token2 with md_Name=mary&md_Score=100
3. Create token3 with md_Name=mary&md_Score=20&md_Grade=A

The following table shows the returned token(s) in case metadata filtering is added in the getUserTokens API:

| Metadata | Token returned |
|----------------------------|---------------------------|
| md_Name=reina | token1 |
| md_Name=mary | token2 and token3 |
| md_Score=100 | token2 |
| md_Name=mary&md_Score=20 | token2 and token3 |
| md_Name=mary&md_Name=reina | token1, token2 and token3 |
| md_Name=* | token1, token2 and token3 |
| md_Uknown=* | Empty list |

For more information on sample curl commands, see [Managing custom Knox Token metadata](#).

Knox SSO Cookie Invalidation

This feature allows a list of pre-configured superusers to invalidate previously issued Knox SSO tokens for (a) particular user(s) in case there is a malicious attack where one (or more) of those users' SSO tokens get compromised.

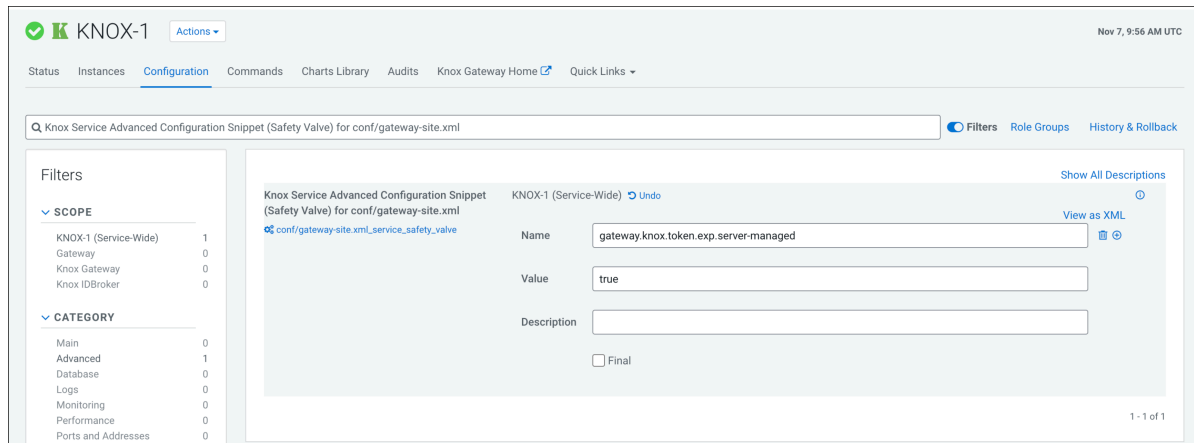
Enabling the feature

By default, the feature is disabled. There are 2 separate steps to enable it:

1. Go to **Cloudera Manager Knox Configuration** and enable **Knox SSO - Cookie Management Enabled**.

The screenshot shows the Cloudera Manager interface for Knox configuration. The main heading is 'KNOX-1' with an 'Actions' dropdown. The 'Configuration' tab is selected, showing a search bar with 'Knox SSO - Cookie Management Enabled'. Below the search bar, there are filters for 'SCOPE' (KNOX-1 (Service-Wide) with a count of 1) and a list of configuration items. The item 'Knox SSO - Cookie Management Enabled' is checked, with sub-items 'knoxssso_cookie_management_enabled' and 'knoxssso_cookie_management_enabled' listed below it. The page also shows 'Nov 7, 9:52 AM UTC' and '1 - 1 of 1'.

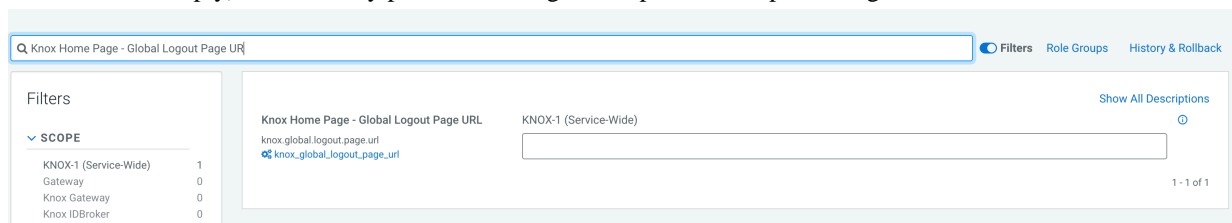
2. In Knox Service Advanced Configuration Snippet (Safety Valve) for `conf/gateway-site.xml`, press +.
 - a. In Name, type `gateway.knox.token.exp.server-managed`.
 - b. In Value, type `true`.
 - c. Click Save Changes(CTRL+S)



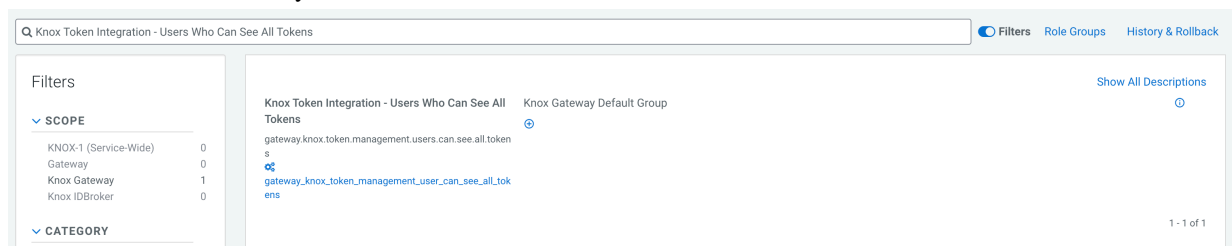
Additional configuration

In addition to enabling the feature, you should review and update the following configuration, if needed:

- Knox Home Page - Global Logout Page URL - when the knoxsso topology is configured to use the Pac4J federation filter (which is the default case in CDP Public Cloud), this configuration is an essential parameter (thus it must not be empty). This usually points to the logout endpoint of the pre-configured SAML/OIDC callback.



- Knox Token Integration - Users Who Can See All Tokens - A comma-separated list of user names who can see all tokens on the Token Management page. By default, this is an empty list. Each organization should configure this property to a narrowed set of users, security officers for instance, who will have the capability of disabling SSO cookies in case of a security breach.



How it works

After enabling the feature, every SSO cookie, the result of a login event through the Knox SSO service, will be recorded in the same database that Knox uses for token management purposes. These SSO cookies are included on the Token Management page. If the logged-in user is a configured "superuser" (added in the above-referenced Users Who Can See All Tokens list), that user is capable of narrowing down user tokens, for whom they suspect are the subject of malicious activities, and disabling the active tokens on the UI (either individually or in batches).

Once a Knox SSO cookie is disabled, it cannot be re-enabled or revoked. Knox has its own cleanup strategy to remove expired tokens from the underlying token state repository (a database in CDP Public Cloud) periodically, on a pre-configured schedule.

It is also important to emphasize, that the default Time To Live (TTL) value of Knox SSO cookies is set to 1 day by default. It's highly recommended that organizations overview their own UI jobs and reduce this value to as short as possible to reduce the security risk involved here.

Concurrent session verification (Tech Preview)

This feature is a security measure that enables end-users limiting the number of concurrent UI sessions the users can have. To achieve this goal the users can be sorted out into three groups: non-privileged, privileged, unlimited.

The non-privileged and privileged groups each have a configurable limit, which the members of the group can not exceed. The members of the unlimited group are able to create an unlimited number of concurrent sessions.

All of the users, who are not configured in either the privileged or in the unlimited group, shall become the member of the non-privileged group by default.



Note: Concurrent session verification feature is under Technical Preview. The technical preview feature and considered under development. Do not use this in your production systems. To share your feedback, contact Support by logging a case on our [Cloudera Support Portal](#). Technical preview features are not guaranteed troubleshooting guidance and fixes.

Configuration

The following table shows the relevant gateway-level parameters that are essential for this feature to work:

| Parameter | Description | Default |
|--|--|---|
| gateway.service.concurrentsessionverifier.impl | To enable the session verification feature, end-users should set this parameter to org.apache.knox.gateway.session.control.InMemoryConcurrentSessionVerifier | org.apache.knox.gateway.session.control.InMemoryConcurrentSessionVerifier |
| gateway.session.verification.privileged.users | Indicates a list of users that are qualified "privileged". | Empty list |
| gateway.session.verification.unlimited.users | Indicates a list of (super) users that can have as many UI sessions as they want. | Empty list |
| gateway.session.verification.privileged.user.limit | The number of UI sessions a "privileged" user can have | 3 |
| gateway.session.verification.non.privileged.user.limit | The number of UI sessions a "non-privileged" user can have | 2 |

How this works

If the verifier is disabled it will not do anything even if the other parameters are configured.

When the verifier is enabled all of the users are considered as a non-privileged user by default and they will not be able to create more concurrent sessions than the non-privileged limit. The same is true after you added someone in the privileged user group: that user will not be able to create more UI sessions than the configured privileged user limit. Whereas the members of the unlimited users group are able to create an unlimited number of concurrent sessions even if they are configured in the privileged group as well.

In Cloudera Data Platform, currently, there are no first-class Cloudera Manager parameters for this feature, so all of those properties have to be set through Knox Service Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml configuration in Cloudera Manager.

KNOX-1 Actions Nov 24, 12:21 PM UTC

Status Instances **Configuration** Commands Charts Library Audits Knox Gateway Home Quick Links

Q gateway-site Filters Role Groups History & Rollback

Filters

SCOPE

| | |
|-----------------------|---|
| KNOX-1 (Service-Wide) | 1 |
| Gateway | 0 |
| Knox Gateway | 2 |
| Knox IDBroker | 1 |

CATEGORY

| | |
|---------------------|---|
| Main | 1 |
| Advanced | 3 |
| Database | 0 |
| Logs | 0 |
| Monitoring | 0 |
| Performance | 0 |
| Ports and Addresses | 0 |
| Resource Management | 0 |

Security - Use FQDN as TLS Certificate Alias Knox Gateway Default Group Show All Descriptions

[gateway_tls_certificate_alias_use_fqdn](#)
[gateway_tls_certificate_alias_use_fqdn](#)

Knox Service Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml KNOX-1 (Service-Wide) Undo View as XML

[conf/gateway-site.xml_service_safety_valve](#)

Name: 🗑️ 🔗

Value:

Description:

Final