Cloudera Runtime 7.1.9

# Apache Knox Authentication

**Date published: 2020-07-28**
**Date modified: 2023-09-07**

**https://docs.cloudera.com/**

# Legal Notice

# Contents

# 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 the Knox Admin UI.

You can access the Knox Admin UI from https://*KNOX_GATEWAY_HOST*:*PORT*/*GATEWAY_PATH*), e.g. https://localhost:8443/gateway/homepage/home.

In the early days of Knox, you enabled Knox proxy by editing topology files manually. Topology files consisted of 3 things:

- Provider configurations: e.g., authentication, federation, authentication, authorization, identity assertion, etc
- HA provider
- Services: component URLs you want to proxy

You configured each of these things in every topology file.

Most recently, topology files are dynamically generated from combinations of Provider Configurations and Descriptors, defined using the Knox Admin UI. Additionally, these provider configurations and descriptors are now shared- you no longer have to specify configurations (e.g. authentication provider, identity assertion provider, or authorization provider) for each topology file- you define a Provider Configuration or Descriptor and they are shared across all topologies you choose. The Admin UI consists of 3 sections:

- Provider Configurations: A named set of providers, e.g., authentication, federation, authentication, authorization, identity assertion, etc. Provider configurations can be shared across descriptors/topologies.
- Descriptors: References the Provider Configurations to declare the policy (authentication, authorization, identity assertion, etc) that goes along with proxying that cluster. Descriptors cannot be shared across topologies; Descriptors and topologies are 1-to-1.
- Topologies: Dynamically generated based on the Provider Configurations and Descriptors you define.

However- the topologies that are managed by Cloudera Manager should be read-only. Within an Cloudera Manager-managed cluster, the Knox Admin UI is to be used for creating additional topologies. When a Knox instance is not managed by Cloudera Manager, all topology management will be done via the Knox Admin UI.

## 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.
- Authorization: Ranger

  OS Security: Data Encryption and HDFS

# 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 | # | # |
| Cloudera Manager UI | # | |

| Component | UI Proxy (with SSO) | API Proxy |
|---|---|---|
| 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 | | |
| Superset | | |

| Component | UI Proxy (with SSO) | API Proxy |
|---|---|---|
| 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.

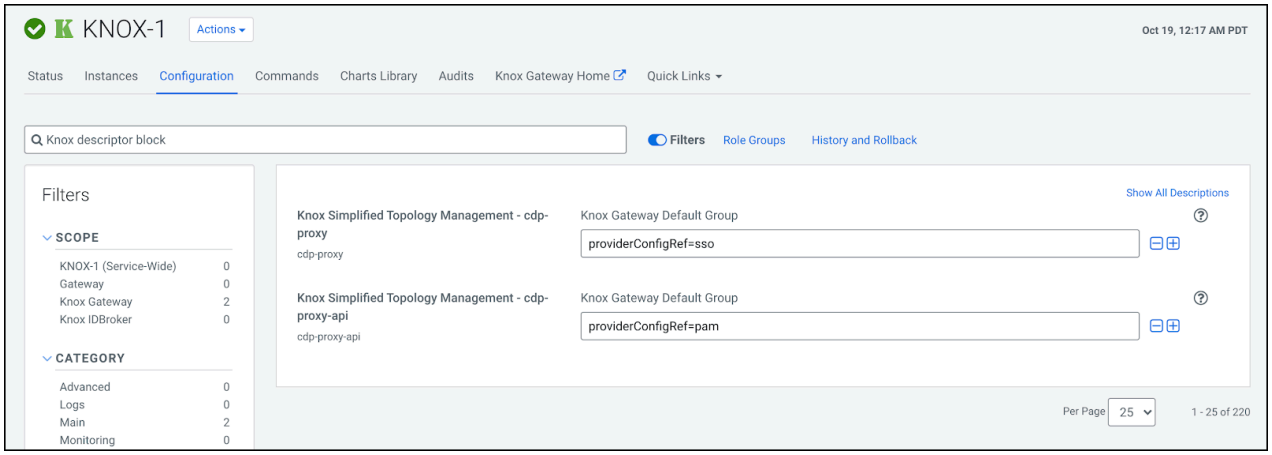
## Knox Topology Management in Cloudera Manager

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 Admin and Home Page UIs.
- 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

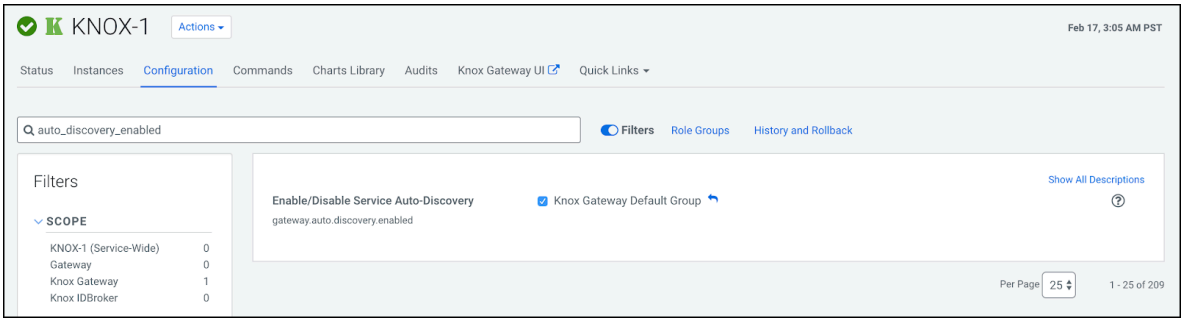
You can enable or disable known or custom services in Knox proxy via Cloudera Manager.

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.

# 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

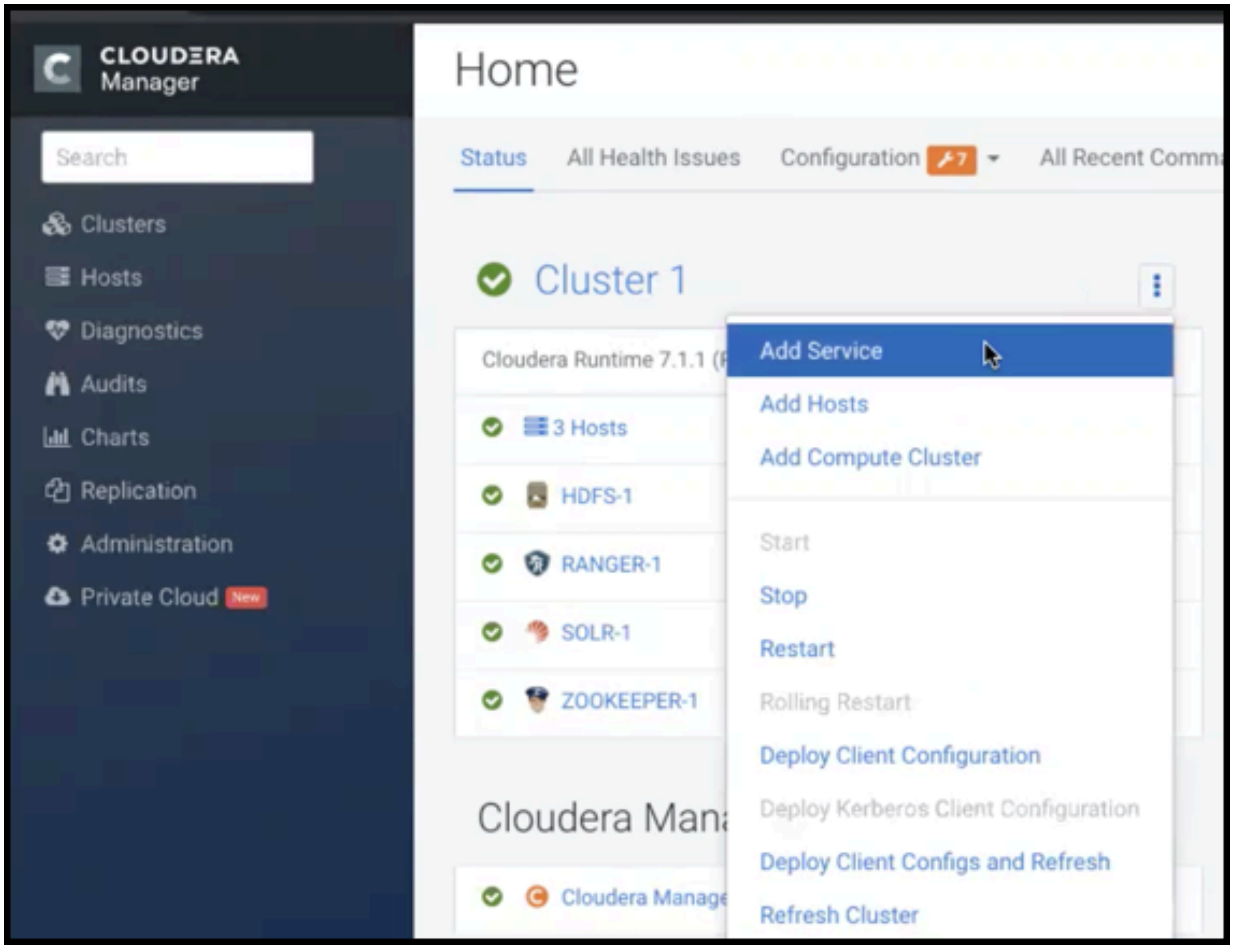
Apache Knox is an application gateway for interacting with the REST APIs and UIs. The Knox Gateway provides a single access point for all REST and HTTP interactions in your Cloudera Data Platform cluster.

## 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 |
| 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 |
|---|---|---|---|
| kerberos.auth.enabled* | Yes | Boolean | false |
| ranger_knox_plugin_hdfs_audit_directory | No | Text | ${ranger_base_audit_url}/knox |
| autorestart_on_stop | No | Boolean | false |
| knox_pam_realm_service | No | Text | login |

| Name | In Wizard | Type | Default Value |
|---|---|---|---|
| save_alias_command_input_password | No | Text | - |

## Knox Gateway role parameters

### Table 3: Required parameters for Knox Gateway role

| Name | In Wizard | Type | Default Value |
|---|---|---|---|
| gateway_master_secret | Yes | Password | - |
| gateway_conf_dir | Yes | Path | /var/lib/knox/gateway/conf |
| gateway_data_dir | Yes | Path | /var/lib/knox/gateway/data |
| gateway_port | No | Port | 8443 |
| gateway_path | No | Text | gateway |
| 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 (Admin UI and Home Page). 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 Admin and Home Page UIs.
- 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.

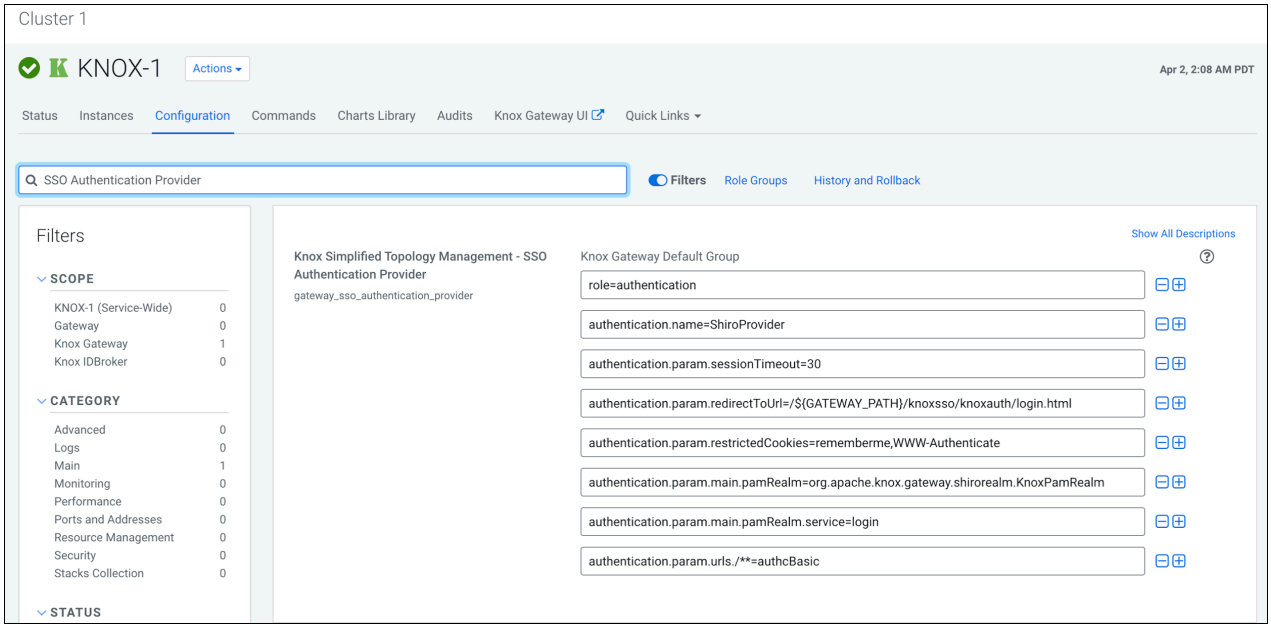
## Configure Apache Knox authentication for PAM

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

### 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}/knoxsso/knoxauth/login.
html
authentication.param.restrictedCookies=rememberme,WWW-Authenticate
authentication.param.urls./**=authcBasic
authentication.param.main.pamRealm=org.apache.knox.gateway.shirorealm.KnoxP
amRealm
authentication.param.main.pamRealm.service=login
```



Every change here goes directly into knoxsso 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.Knox
PamRealm
authentication.param.main.pamRealm.service=login
```

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

## Configure Apache Knox authentication for AD/LDAP

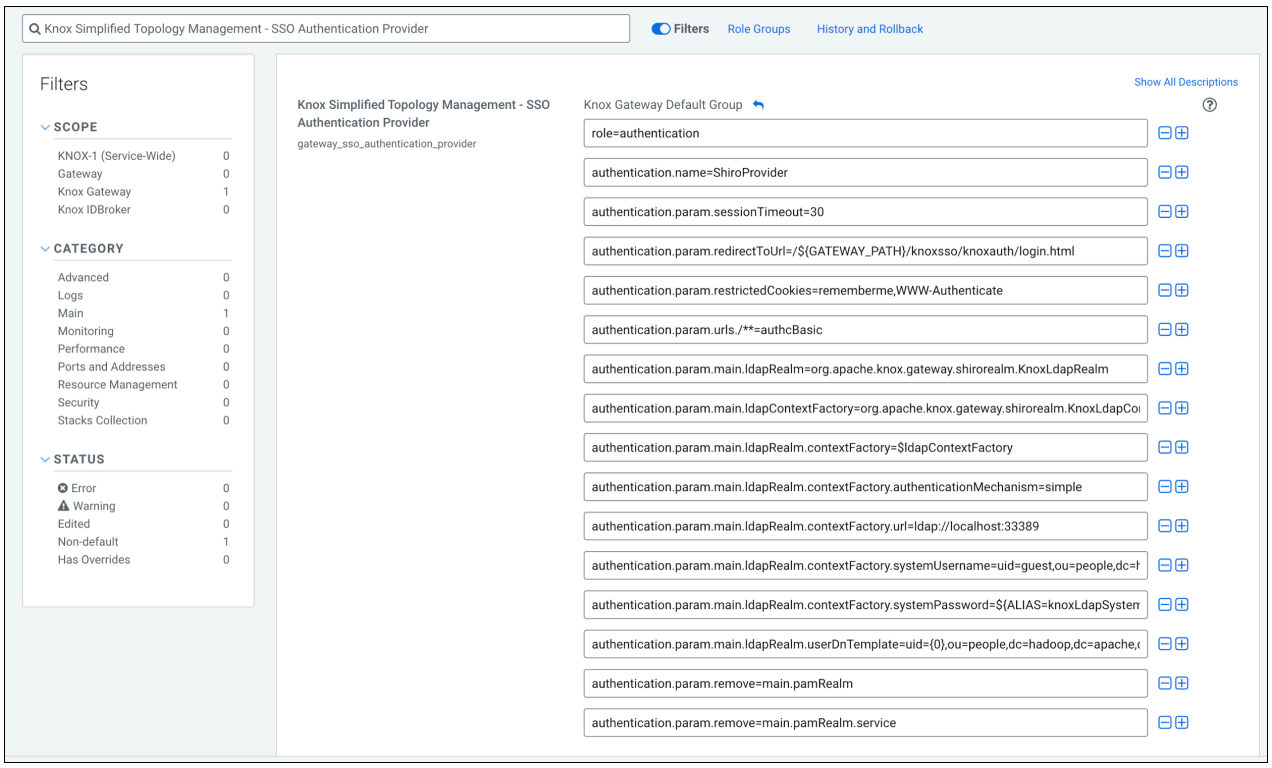
Knox authentication configurations for LDAP and AD in Cloudera Manager.

### 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=/${GATEWAY_PATH}/knoxsso/knoxauth/login.
html
authentication.param.restrictedCookies=rememberme,WWW-Authenticate
authentication.param.urls./**=authcBasic
authentication.param.main.ldapRealm=org.apache.knox.gateway.shirorealm.Knox
LdapRealm
authentication.param.main.ldapContextFactory=org.apache.knox.gateway.shiro
realm.KnoxLdapContextFactory
authentication.param.main.ldapRealm.contextFactory=$ldapContextFactory
authentication.param.main.ldapRealm.contextFactory.authenticationMechanism=s
imple
authentication.param.main.ldapRealm.contextFactory.url=ldap://localhost:33
389
authentication.param.main.ldapRealm.contextFactory.systemUsername=uid=guest,
ou=people,dc=hadoop,dc=apache,dc=org
authentication.param.main.ldapRealm.contextFactory.systemPassword=${ALIAS=k
noxLdapSystemPassword}
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,D
C=COM
authentication.param.main.ldapRealm.userObjectClass=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.




**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 knoxui:knoxui 'https://johndoe-1.abc.cloudera.com:8443/gateway/ad
min/api/v1/providerconfig/knoxsso'
...
  }, {
    "role" : "authentication",
    "name" : "ShiroProvider",
    "enabled" : true,
    "params" : {
      "main.ldapContextFactory" : "org.apache.knox.gateway.shirorealm.Knox
LdapContextFactory",
      "main.ldapRealm" : "org.apache.hadoop.gateway.shirorealm.KnoxLdapReal
m",
      "main.ldapRealm.contextFactory" : "$ldapContextFactory",
      "main.ldapRealm.contextFactory.authenticationMechanism" : "simple",
      "main.ldapRealm.contextFactory.systemPassword" : "${ALIAS=knoxLdapSys
temPassword}",
      "main.ldapRealm.contextFactory.systemUsername" : "uid=guest,ou=peop
le,dc=hadoop,dc=apache,dc=org",
      "main.ldapRealm.contextFactory.url" : "ldap://localhost:33389",
      "main.ldapRealm.userDnTemplate" : "uid={0},ou=people,dc=hadoop,dc=ap
ache,dc=org",
      "redirectToUrl" : "/${GATEWAY_PATH}/knoxsso/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.Kno
xLdapRealm
authentication.param.main.ldapContextFactory=org.apache.knox.gateway.shir
orealm.KnoxLdapContextFactory
authentication.param.main.ldapRealm.contextFactory=$ldapContextFactory
authentication.param.main.ldapRealm.contextFactory.authenticationMechanism=
simple
authentication.param.main.ldapRealm.contextFactory.url=ldap://localhost:3
3389
authentication.param.main.ldapRealm.contextFactory.systemUsername=uid=guest
,ou=people,dc=hadoop,dc=apache,dc=org
authentication.param.main.ldapRealm.contextFactory.systemPassword=${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.userObjectClass=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.

## 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/knoxsso/api/v1/websso federation.para
m.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.shi
rorealm.KnoxLdapContextFactory#
authentication.param.main.ldapRealm.contextFactory=$ldapContextFactory#
authentication.param.main.ldapRealm.contextFactory.authenticationMechani
sm=simple#
authentication.param.main.ldapRealm.contextFactory.url=ldap://ldap-ho
st: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 Admin UI (https://*KNOX_GATEWAY_HOST*:*PORT*/*GATEWAY_PATH*/gateway/manager/admin-ui/), navigate to the Provider Configurations, and verify that your provider configuration was generated with the providers and parameters 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 Clouder 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 5: Default shared provider configurations

| Configuration | Used by these topologies |
|---|---|
| admin | admin |
| homepage | homepage |
| knoxsso | homepage<br>cdp-proxy<br>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.

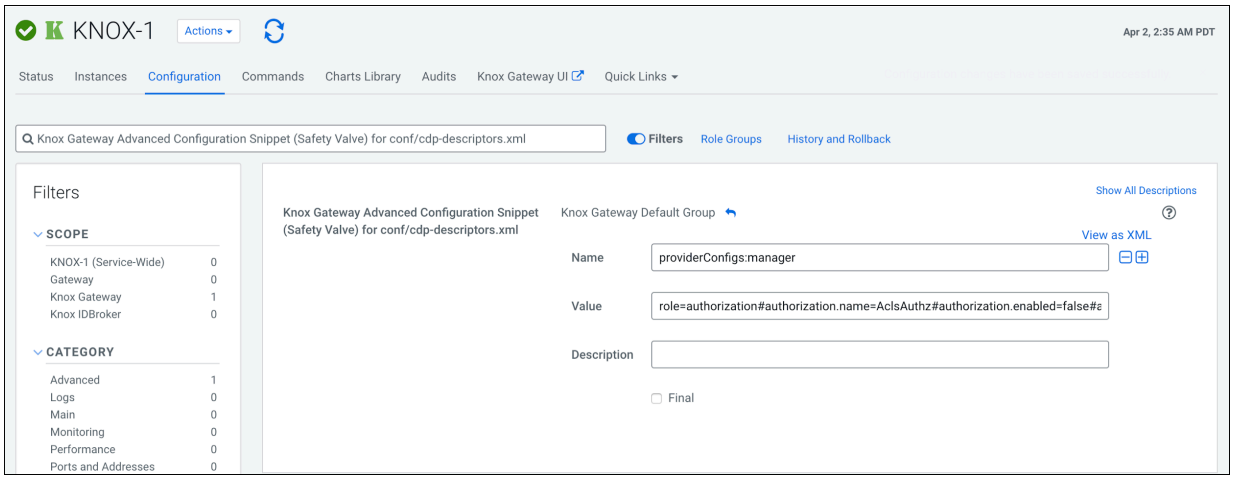
### About this task

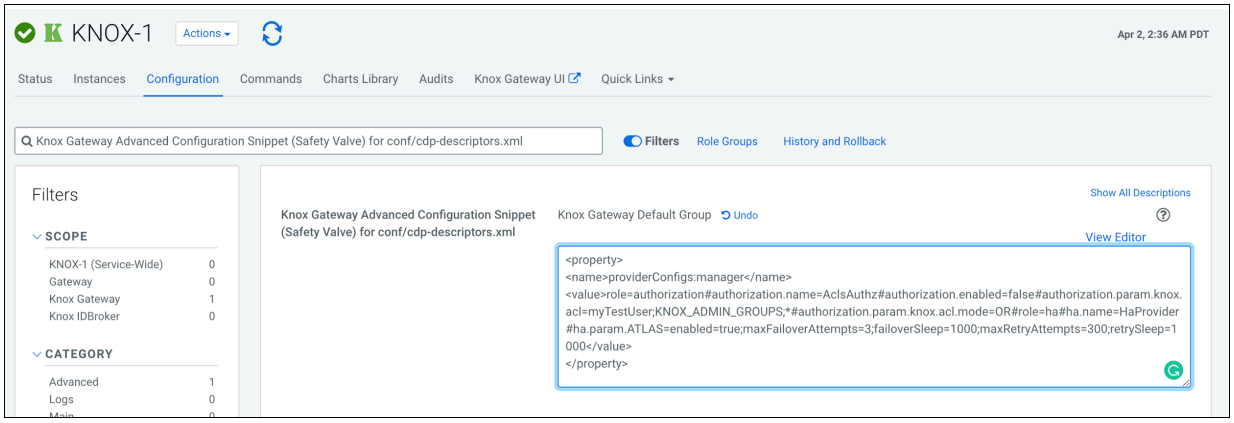
In this example you will see how to add a new HA provider (this time only the ATLAS service will be configured for high availability) in the manager shared provider configuration . This particular authorization provider is set as follows (in its JSON descriptor):

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

## Procedure

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: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;maxRetr




2. Save your changes.
3. Refresh the cluster.
4. Validate:

```
$ curl -ku knoxui:knoxui '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;m
axRetryAttempts=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.

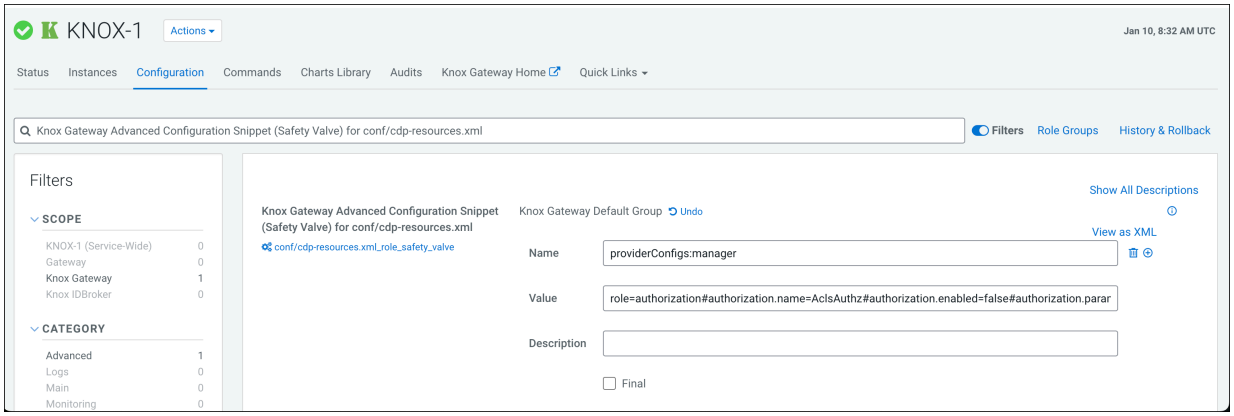
### About this task

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

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:manager
   - value = role=authorization#authorization.name=AclsAuthz#authorization.enabled=false#authorization.param.knox.acl=myTestUser;KNOX_ADMIN_GROUPS;*#authorization.param.knox.acl.mode=OR




   With this change you are authorizing a user called myTestUser to login and execute administrative actions on the Knox Admin UI.
2. Save your changes.
3. Refresh the cluster.
4. Validate:

```
$ curl -ku knoxui:knoxui '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;m
axRetryAttempts=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.

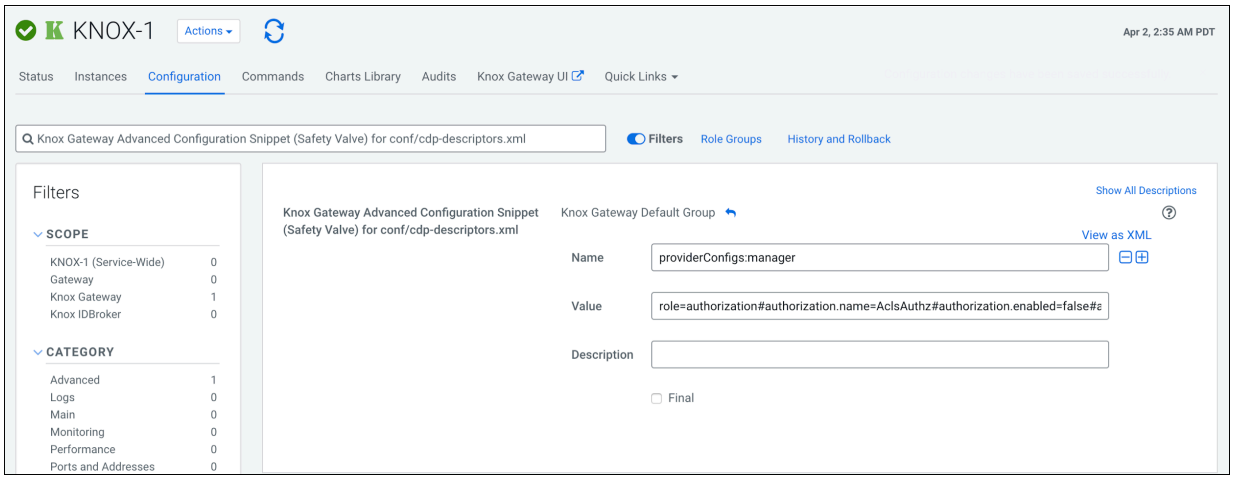
### About this task

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

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: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




2. Save your changes.
3. Refresh the cluster.
4. Validate:

```
$ curl -ku knoxui:knoxui '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;m
axRetryAttempts=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 knoxui:knoxui '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

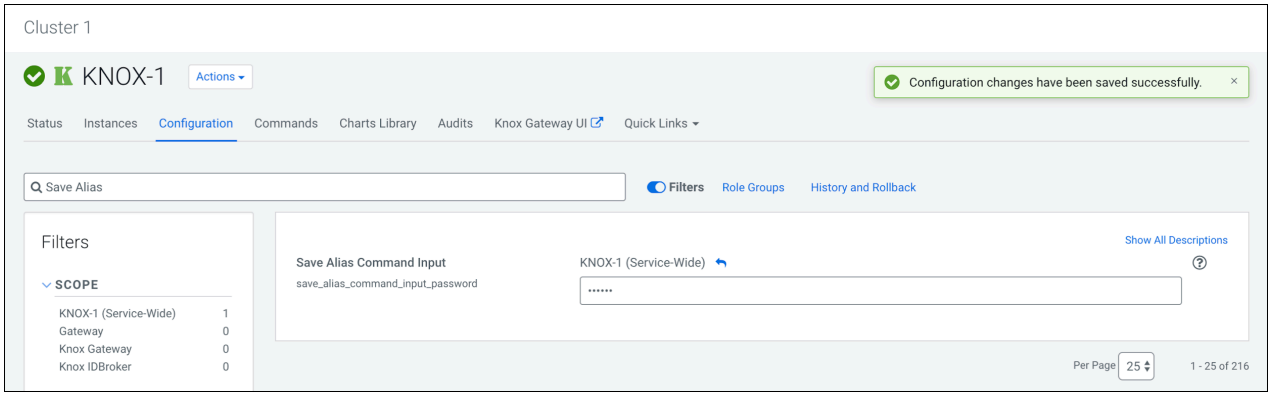
There is a new command available for the Knox Gateway role which allows end-users to save an alias=password pair to an arbitrary number of topologies on each host where an instance of the Knox Gateway is installed without the need of running the Knox CLI tool manually.

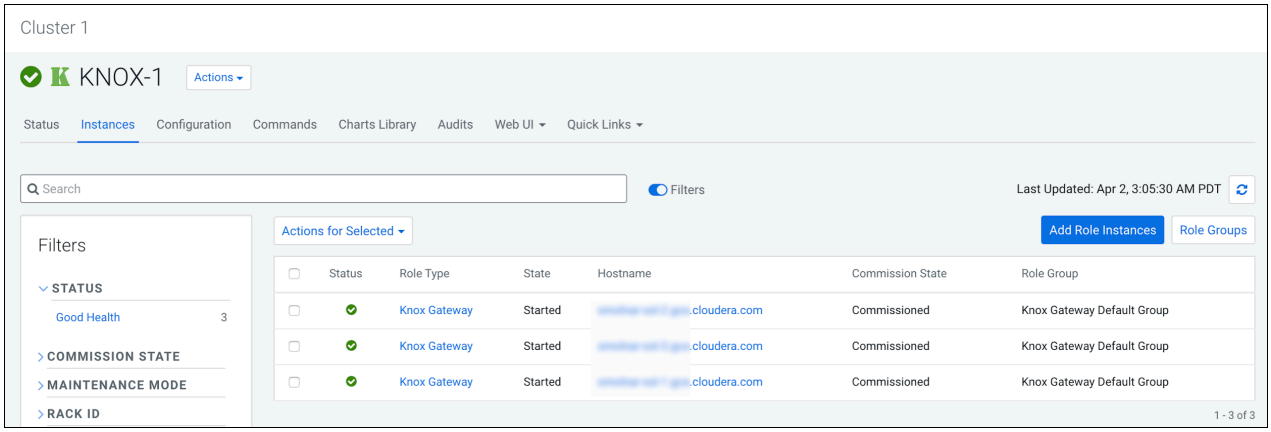
A new password-type input field is added, called save_alias_command_input_password. The format of an entry in this input field should be: topology_name_1[:topology_name_2:...:topology_name_N].alias_name=password

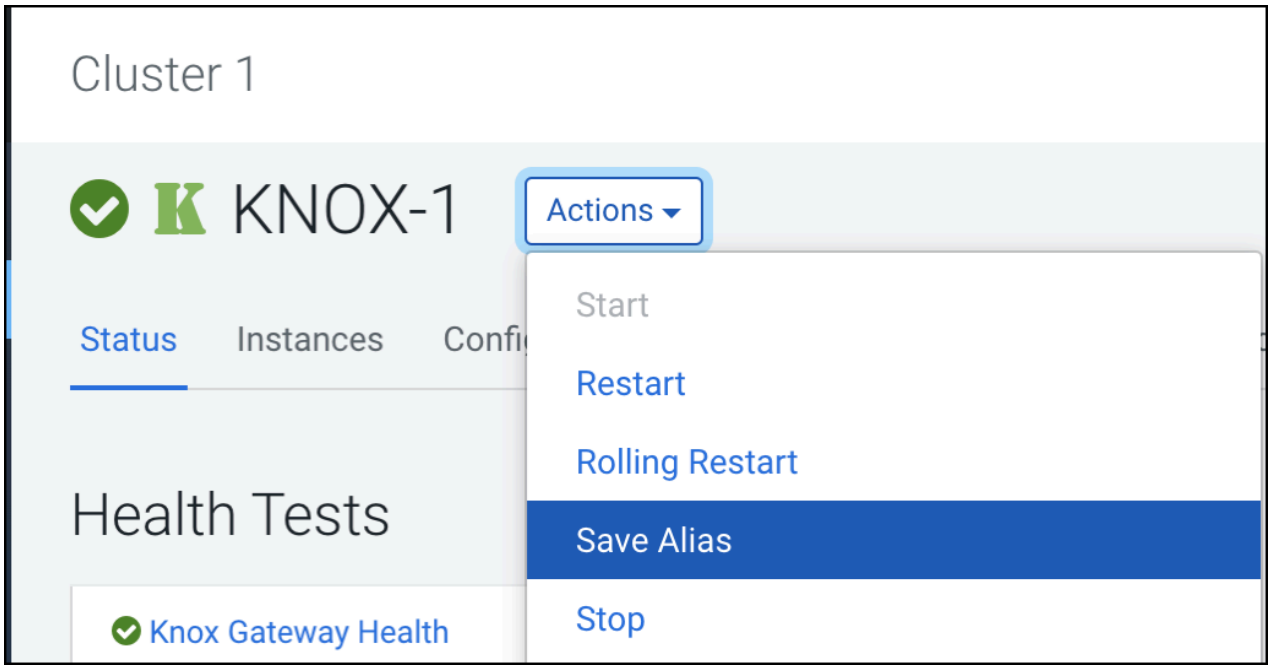
Example: cdp-proxy-api:admin:metadata.knoxLdapSystemPassword=guest-password.

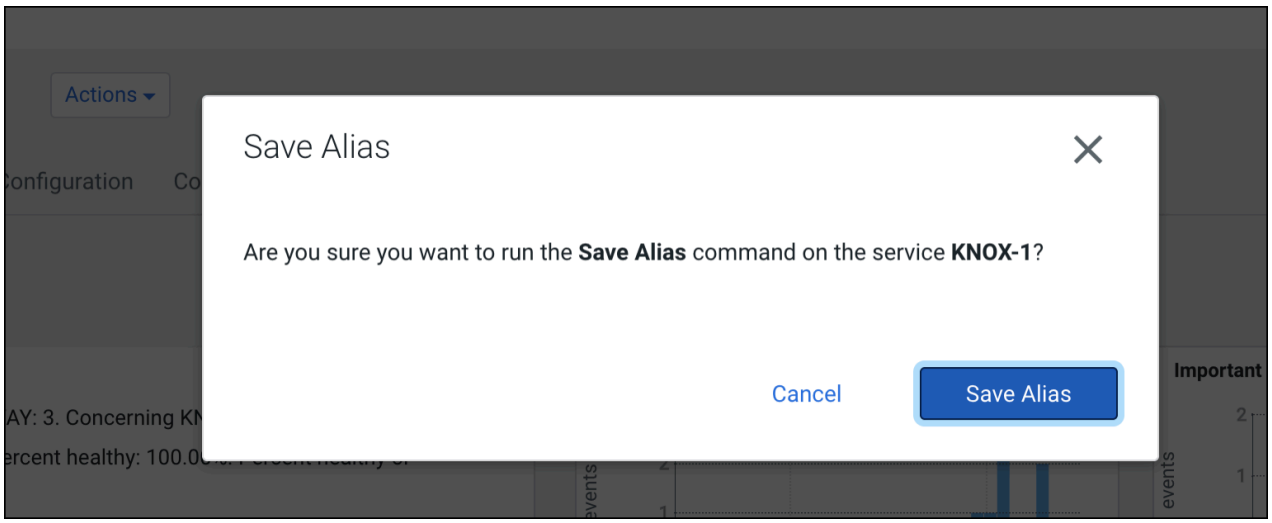
After the end-users entered a meaningful and valid value and saved the configuration changes they can run the command from Knox’s action list: Actions/Save Alias.

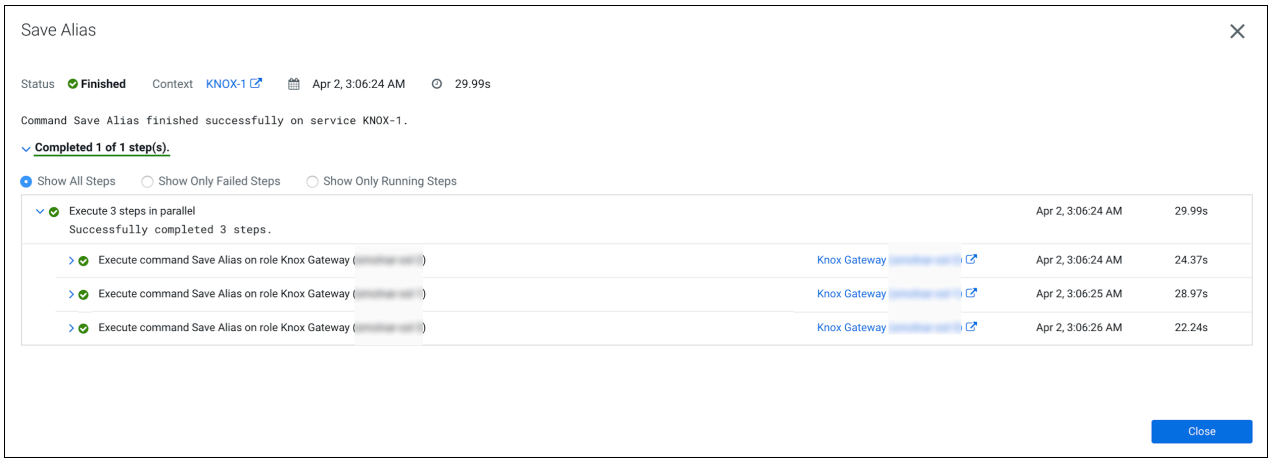
**Tip:** If you need to add a Gateway level alias, please use __gateway as topology name. For instance: __gateway.knoxLdapSystemPassword=admin-password.

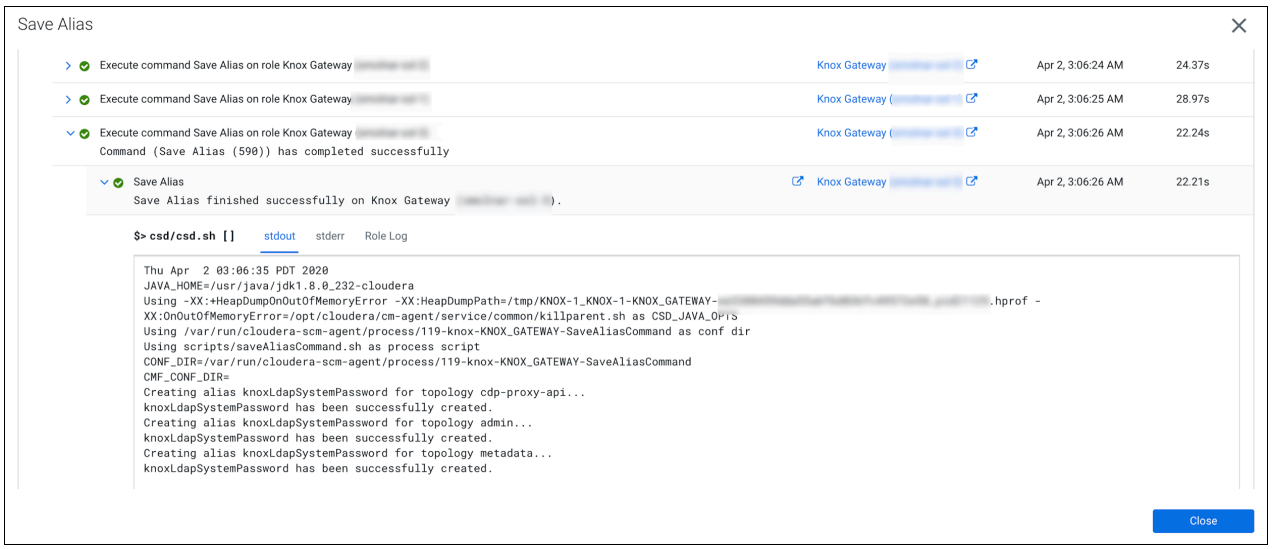





## Configuring Kerberos authentication in Apache Knox shared providers

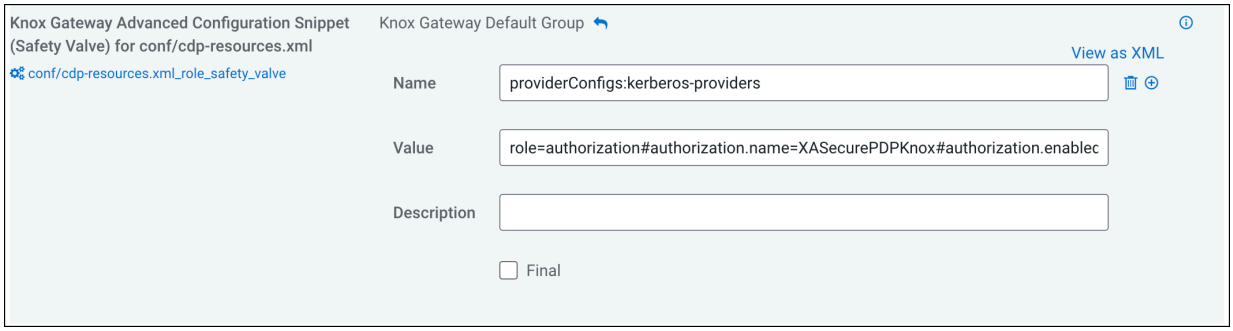
An example of how to add the kerberos-auth configuration provider from Cloudera Manager.

## 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 =

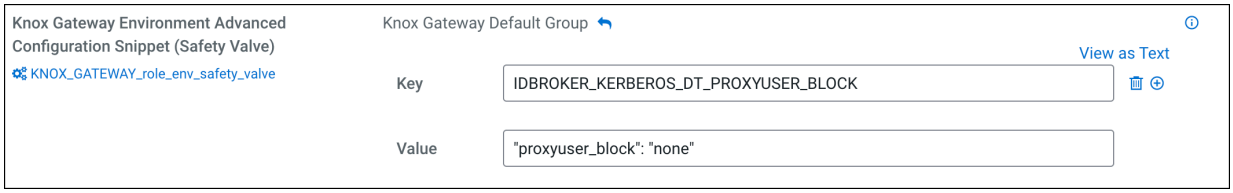
```
role=authorization#
authorization.name=XASecurePDPKnox#
authorization.enabled=true#
role=ha#
ha.name=HaProvider#
ha.enabled=true#
ha.param.HBASE=maxFailoverAttempts=3;failoverSleep=1000;enabled=true#
ha.param.HIVE=maxFailoverAttempts=3;failoverSleep=1000;enabled=true;
zookeeperEnsemble=maxFailoverAttempts=3;failoverSleep=1000;enabled=true;
zookeeperEnsemble=gbl20175161.systems.uk.company:2181,gbl20175162.syst
ems.uk.company:2181,gbl20175163.systems.uk.company:2181;
zookeeperNamespace=hiveserver2#
ha.param.OOZIE=maxFailoverAttempts=3;failoverSleep=1000;enabled=true#
ha.param.WEBHCAT=maxFailoverAttempts=3;failoverSleep=1000;enabled=true#
ha.param.WEBHDFS=maxFailoverAttempts=3;failoverSleep=1000;maxRetryAtt
empts=300;retrySleep=1000;enabled=true
```

**Important:** Paste the value = code as a single line, for example: role=authorization#authorization .name=XASecurePDPKnox#[...]# ha.param.WEBHDFS=maxFailoverAttempts=3;failoverSlee p=1000;maxRetryAttempts=300;retrySleep=1000;enabled=true




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"
```




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/providerc
onfig/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_SIGNA
TURE_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

# Management of services for Apache Knox via Cloudera Manager

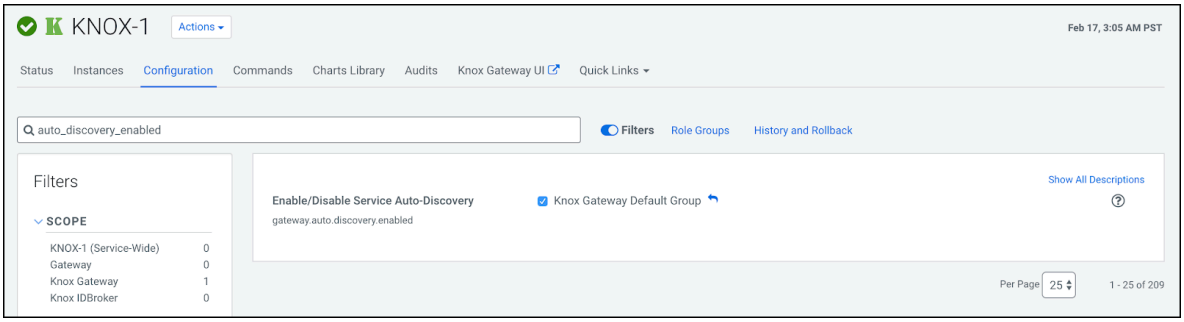
You can enable or disable known or custom services in Knox proxy via Cloudera Manager.

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) 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:




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.

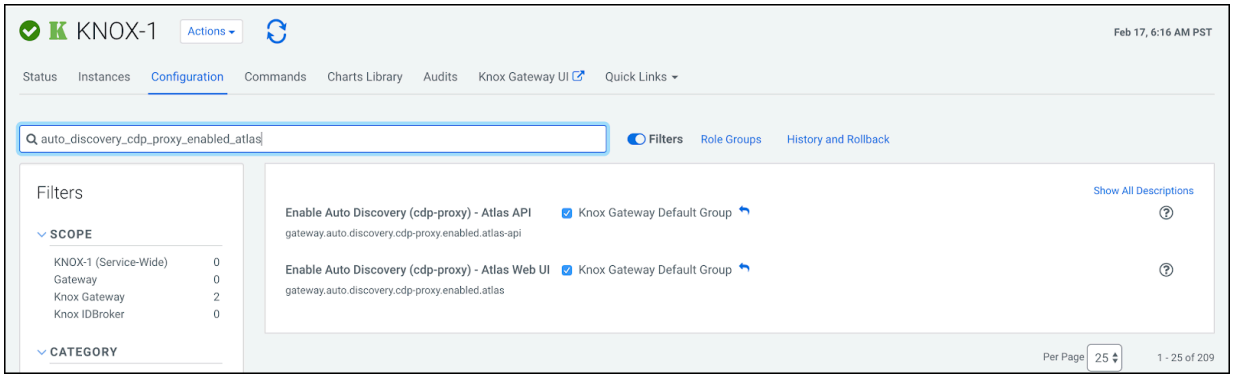
### About this task

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

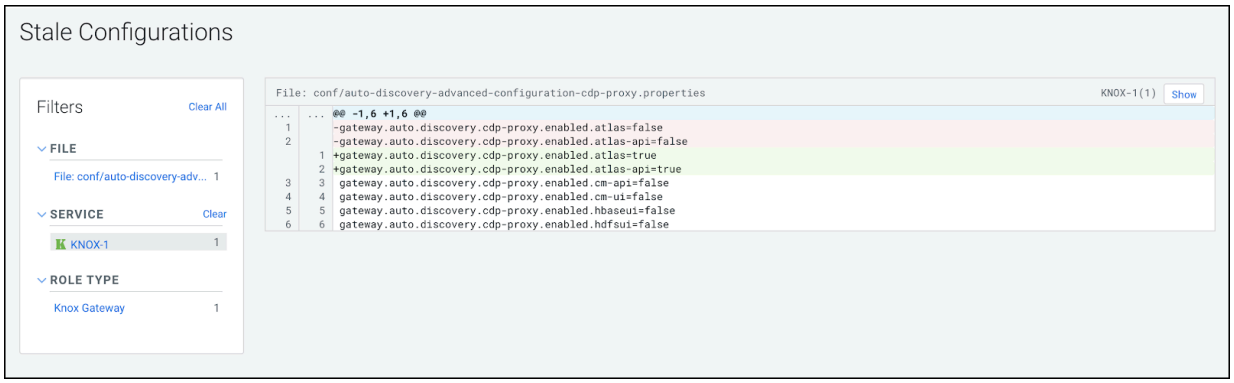
For the purposes of this example, we add ATLAS and ATLAS UI to cdp-proxy. You can add more services; for a comprehensive list of knoxn services that can be enabled, see "Knox Supported Services Matrix".

### Procedure

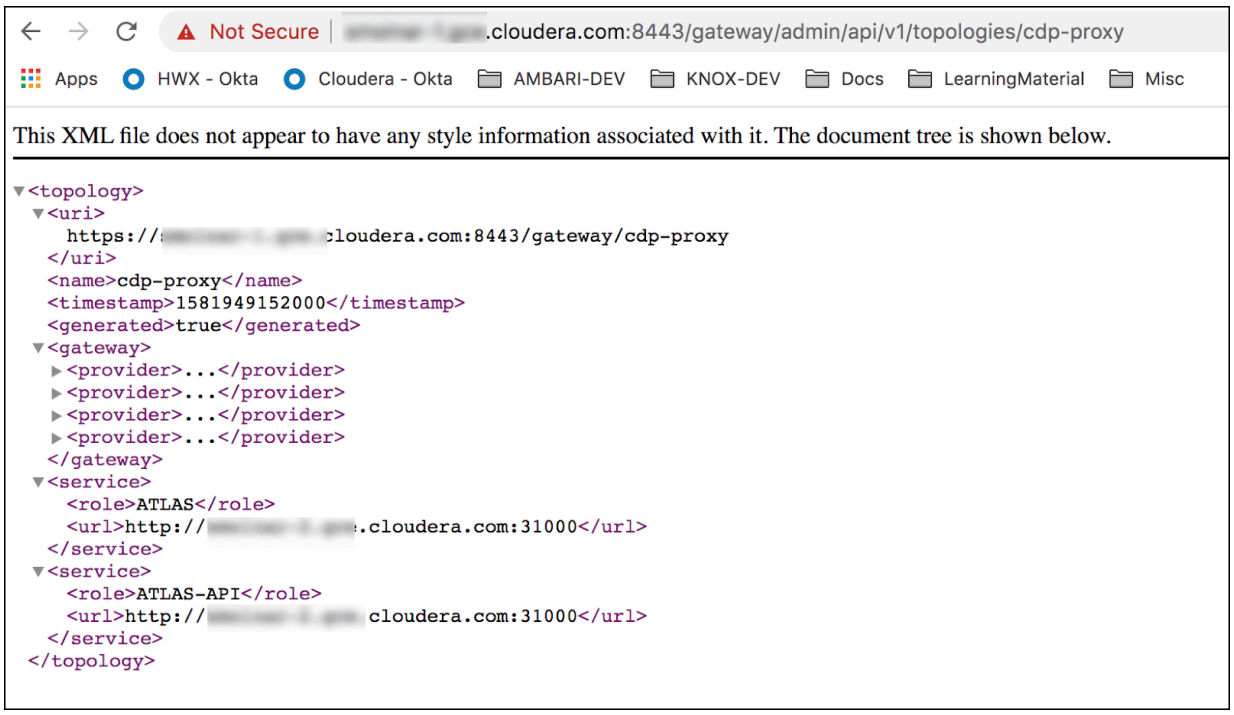
1. From Cloudera Manager Knox Configuration , check the Gateway Auto Discovery (cdp-proxy) - $Component boxes.
   In this example, we enable:
   - gateway_auto_discovery_cdp_proxy_enabled_atlas
   - gateway_auto_discovery_cdp_proxy_enabled_atlas_ui



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




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




### 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.

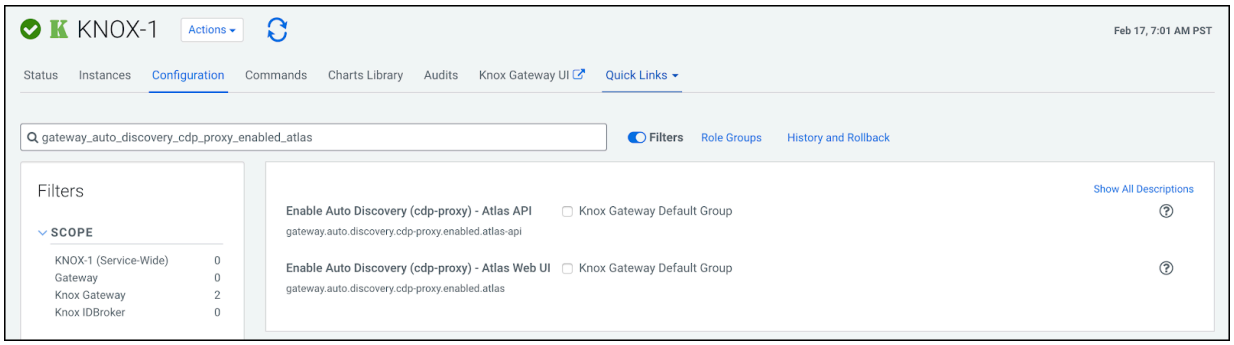
### About this task

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

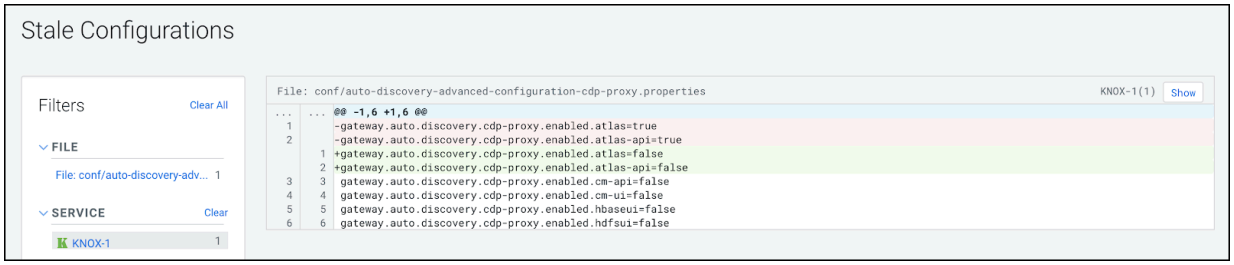
In this example, we are going to remove the previously added ATLAS and ATLAS-UI services from cdp-proxy. We disable the gateway_auto_discovery_cdp_proxy_enabled_atlas and gateway_auto_discovery_cdp_proxy_enabled_atl as_ui checkboxes on Knox’s Configuration page in CM, save the changes and refresh the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

### Procedure

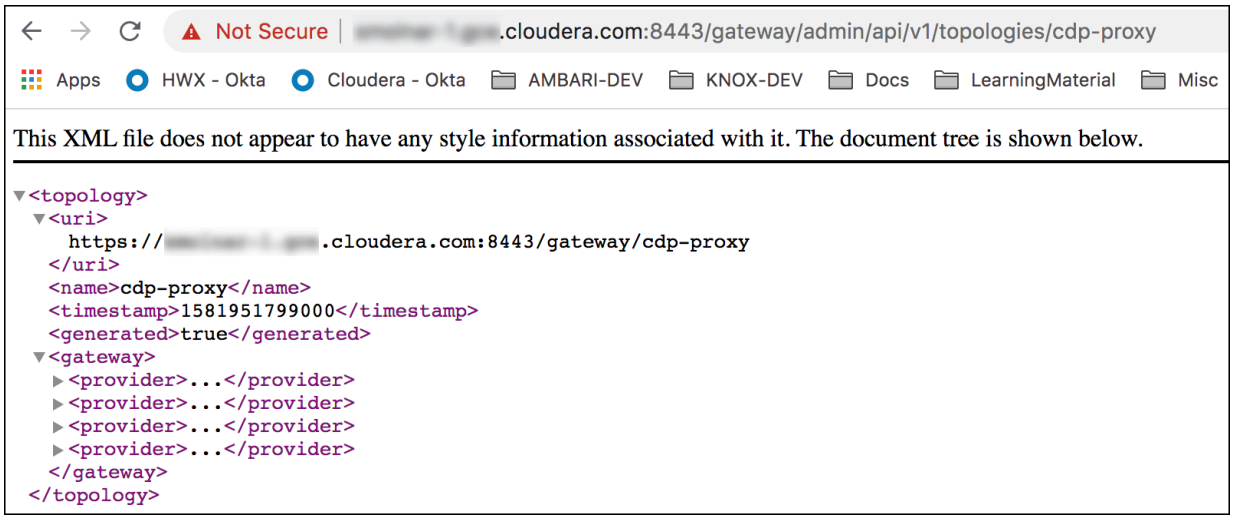
1. From Cloudera Manager Knox Configuration , uncheck the Gateway Auto Discovery (cdp-proxy) - $Component boxes.
   In this example, we disable:
   - gateway_auto_discovery_cdp_proxy_enabled_atlas
   - gateway_auto_discovery_cdp_proxy_enabled_atlas_ui




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




4. Validate that custom service got removed by going to the following URL: http s://*$KNOX_GATEWAY_HOST*:*$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.

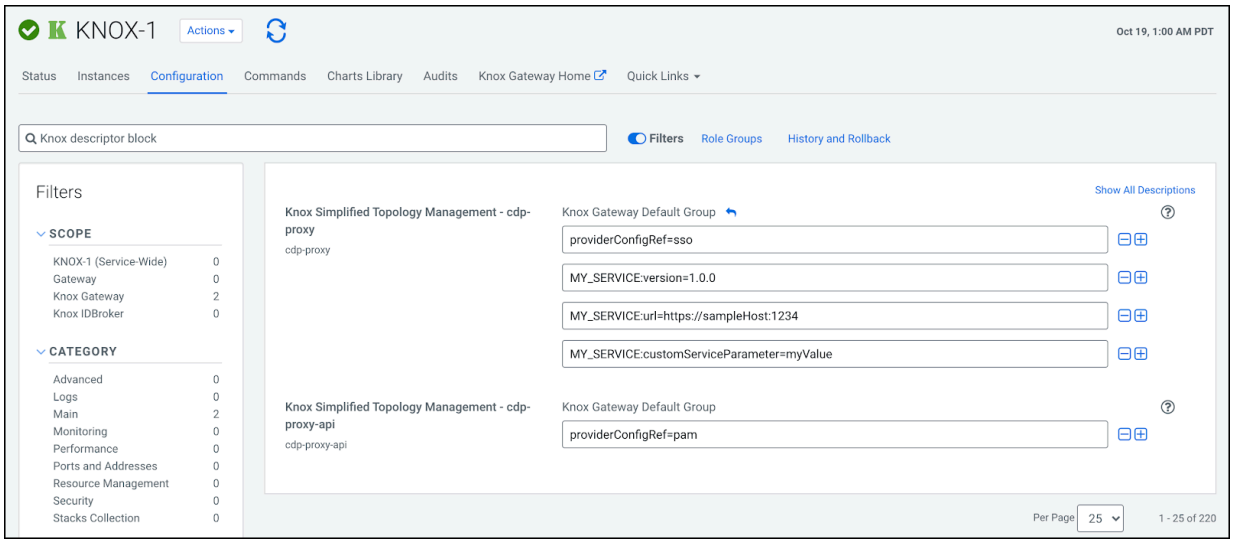
**Important:** Adding a custom service only works if you provide the service definition files (service.xml and rewrite.xml) in the KNOX_DATA_DIR/services folder.

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.

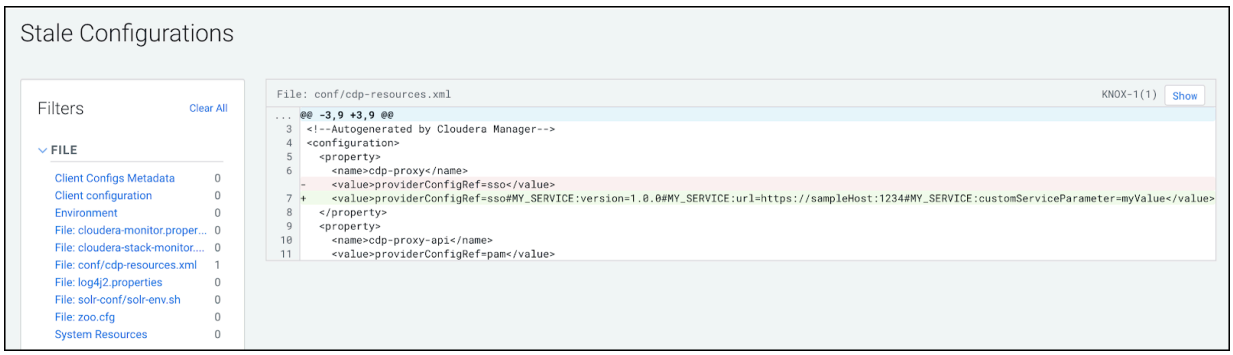
## Procedure

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 s://*$KNOX_GATEWAY_HOST*:*$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 Configurat ion 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.

3. Refresh the Knox instances' configuration: the Refresh needed stale configuration indicator appears; click it and wait until the refresh process completes.
4. Validate:

   Using the Knox Admin UI (https://*KNOX_GATEWAY_HOST*:*PORT*/*GATEWAY_PATH*/gateway/manager/admin-ui/), navigate to the Topologies, and verify that your topology was 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

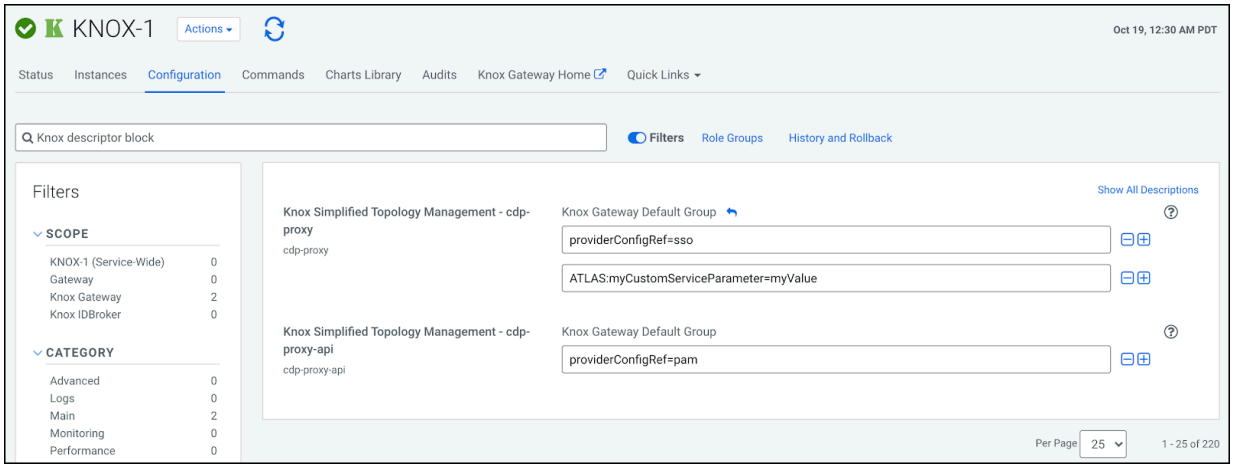
In this example, you are adding a custom service parameter with a custom value (myCustomServiceParameter=myValue) to ATLAS in cdp-proxy.

### Procedure

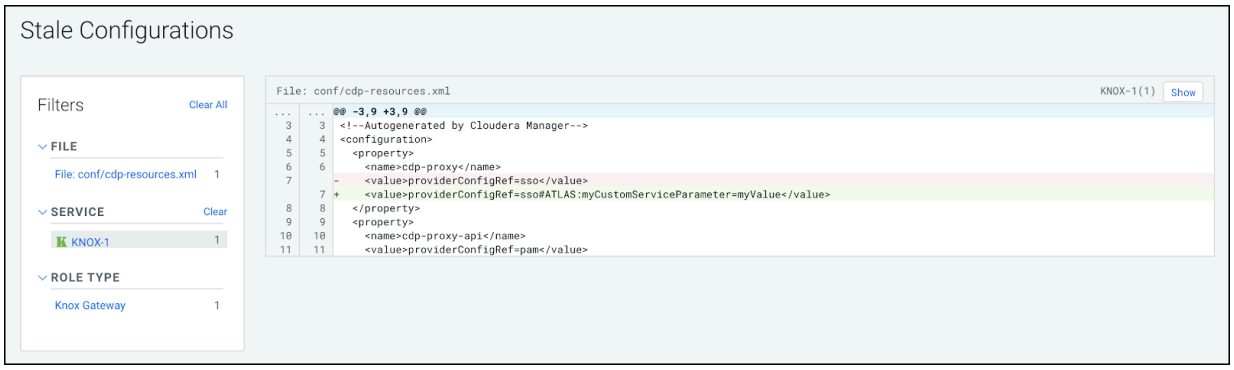
1. From Cloudera Manager Knox Configuration , add a new line in the Knox Simplified Topology Management - cdp-proxy panel in the following format:
   $*SERVICE_NAME*[:$*PARAMETER_NAME*=$*PARAMETER_VALUE*].
   ATLAS:myCustomServiceParameter=myValue

   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
```

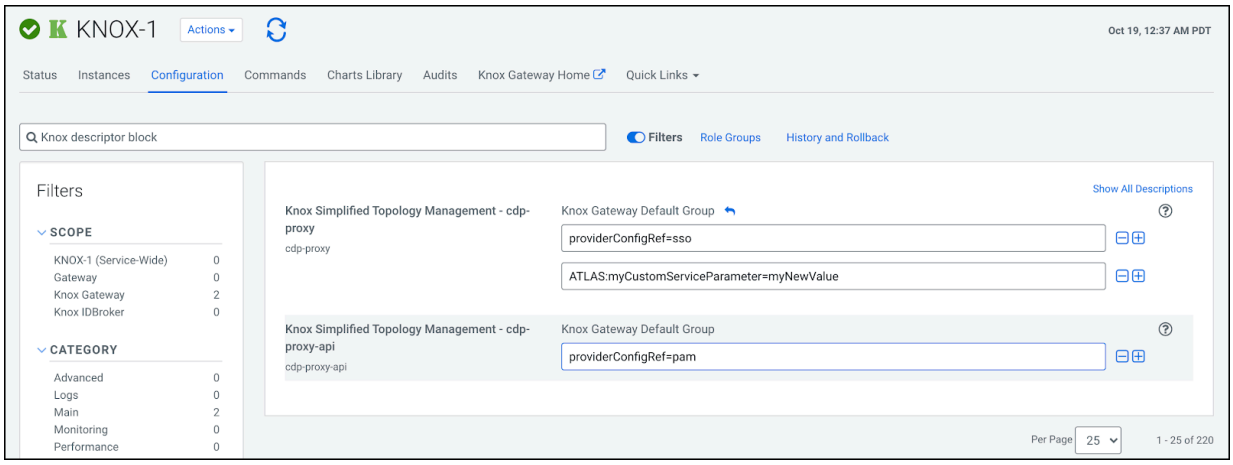


2. Save your changes.
3. The ‘Refresh needed’ stale configuration indicator appears; click it and wait until the refresh process completes.

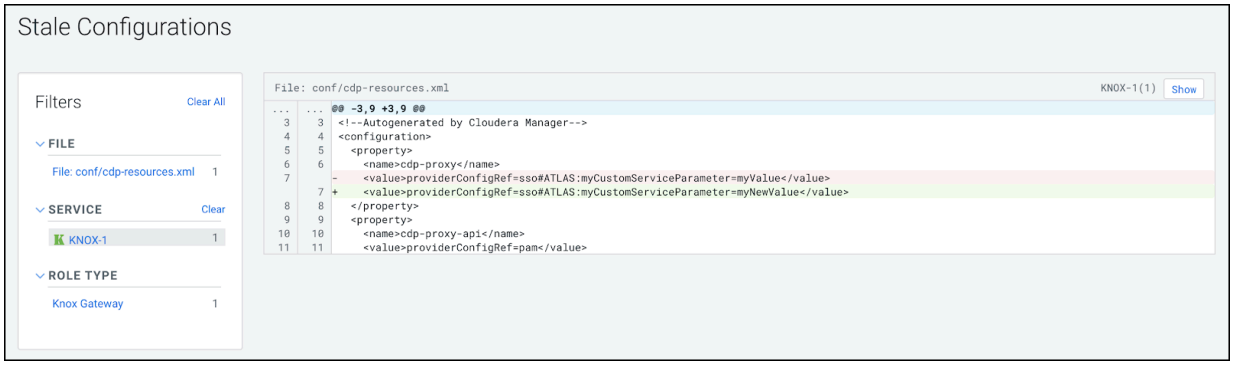

Stale Configurations
Filters
Clear All
FILE
File: conf/cdp-resources.xml 1
SERVICE
Clear
KNOX-1 1
ROLE TYPE
Knox Gateway 1
File: conf/cdp-resources.xml
KNOX-1(1)
Show
@@ -3,9 +3,9 @@
<!--Autogenerated by Cloudera Manager-->
<configuration>
<property>
<name>cdp-proxy</name>
<value>providerConfigRef=sso</value>
<value>providerConfigRef=sso#ATLAS:myCustomServiceParameter=myValue</value>
</property>
<property>
<name>cdp-proxy-api</name>
<value>providerConfigRef=pam</value>

## Procedure

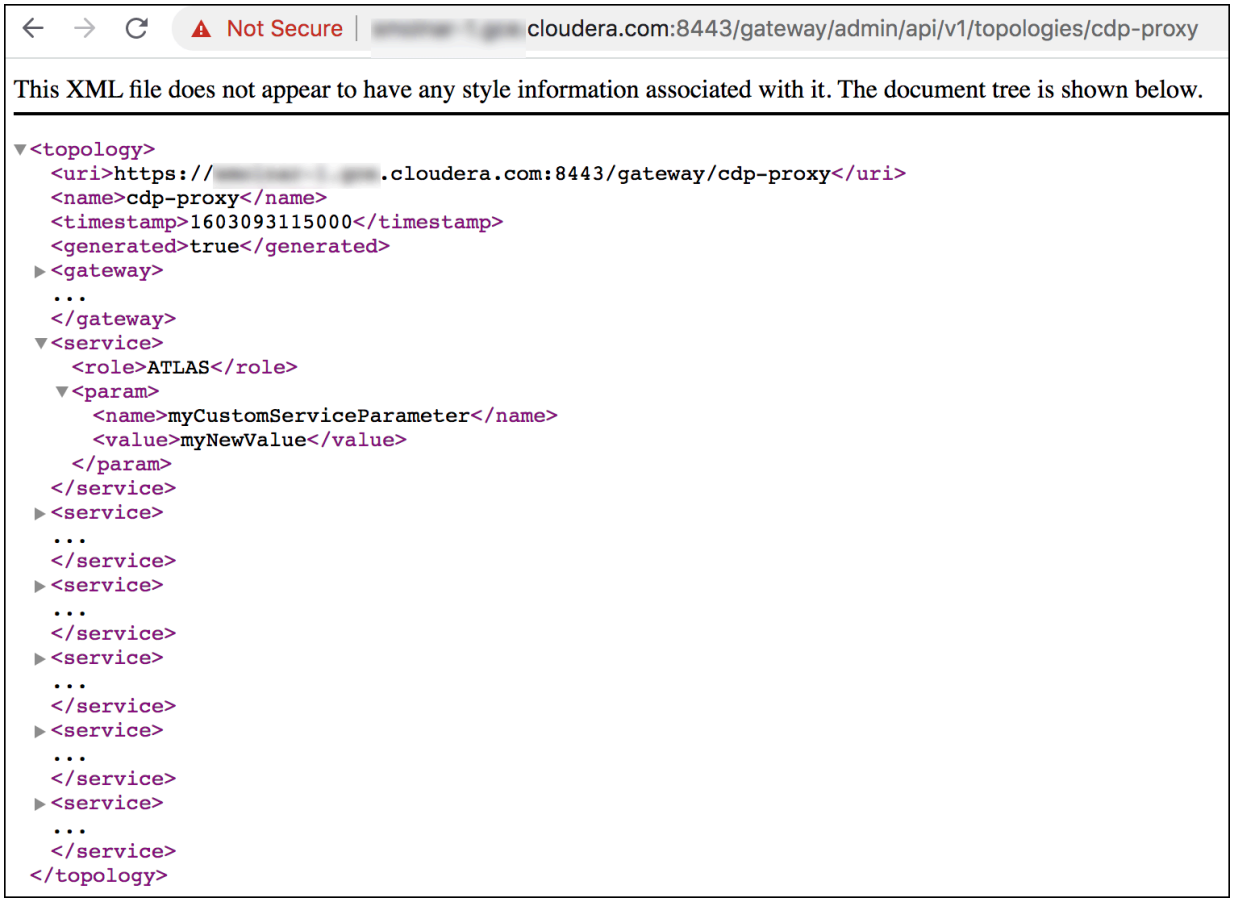
1. From Cloudera Manager Knox Configuration , change the service parameter in the Knox Simplified Topology Management - cdp-proxy panel.
   Change ATLAS:myCustomServiceParameter=myValue to Atlas:myCustomServiceParameter=myNewValue




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



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




## Remove custom service parameter from descriptor

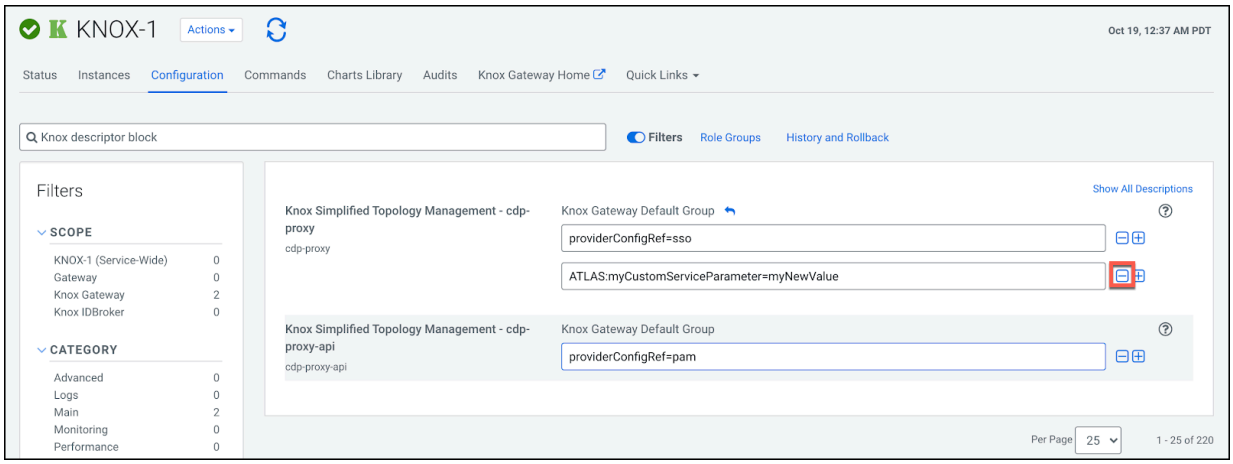
How to remove a custom service parameter from a descriptor using Cloudera Manager.

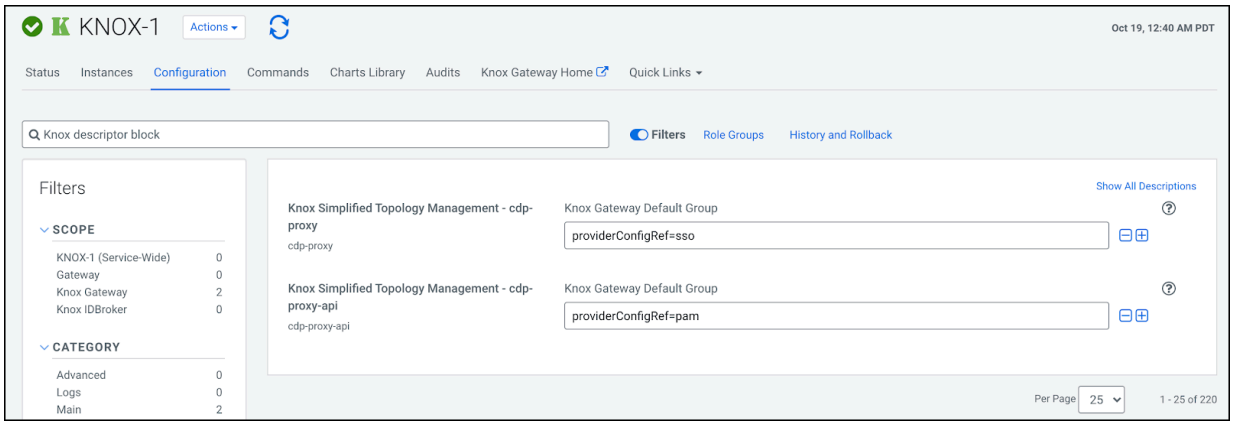
### About this task

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

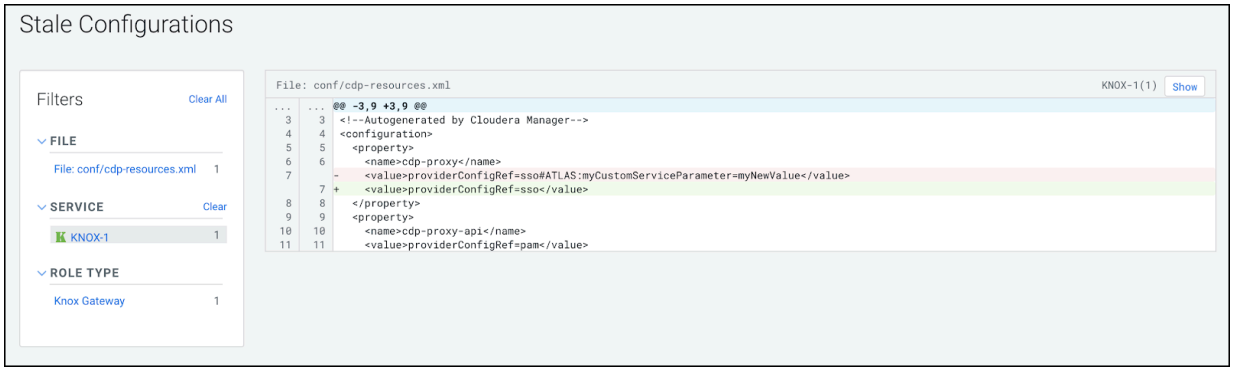
## Procedure

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

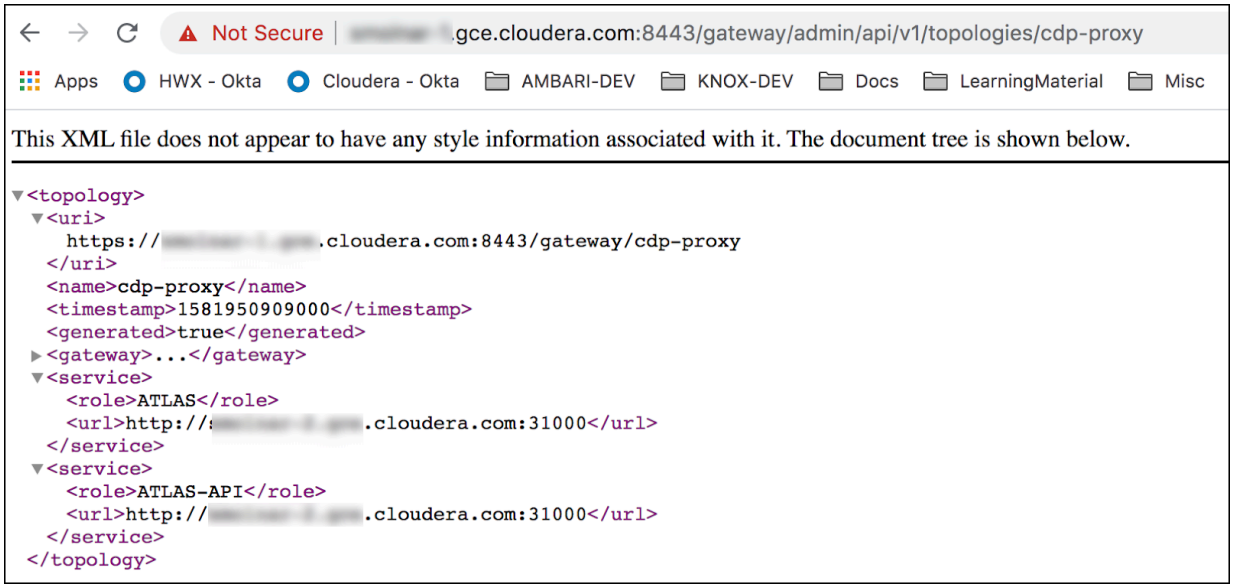



2. Save your changes.
3. The ‘Refresh needed’ stale configuration indicator appears; click it and wait until the refresh process completes.



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.




# 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 (see Manually Configuring TLS Encryption for Cloudera Manager). Then copy to /var/lib/knox/gateway/data/security/keystores/.
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.passwor d.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).

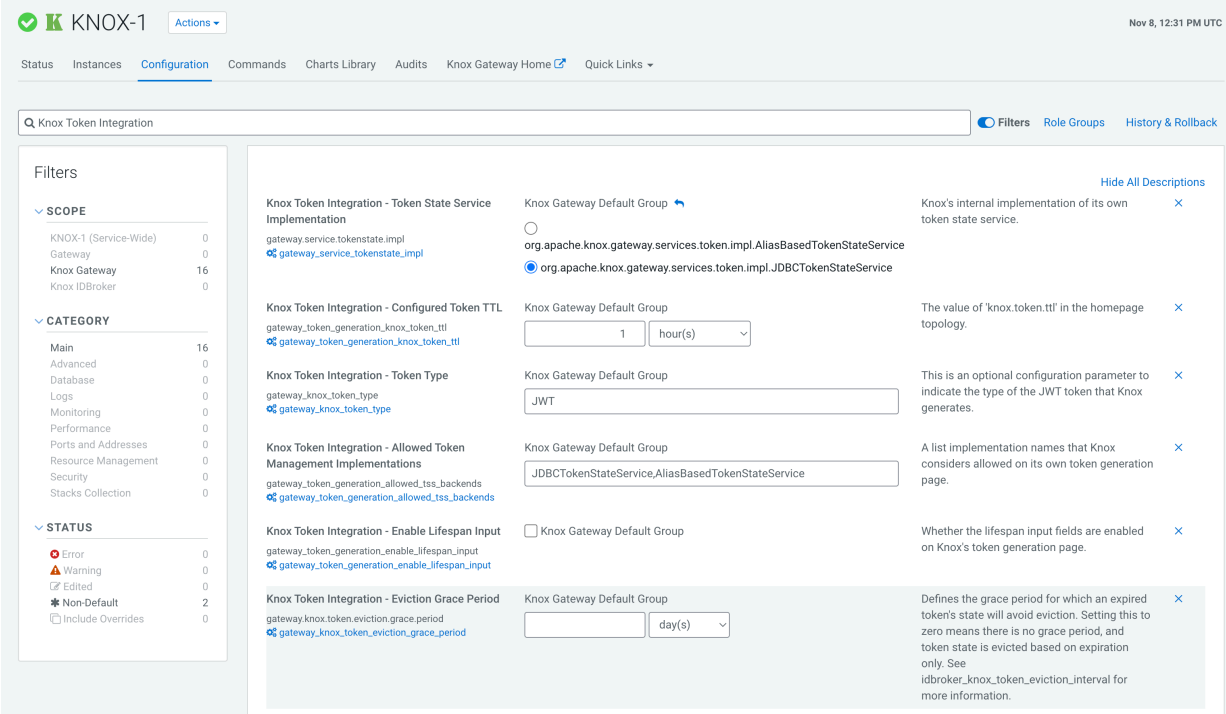
## About Knox gateway tokens

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.

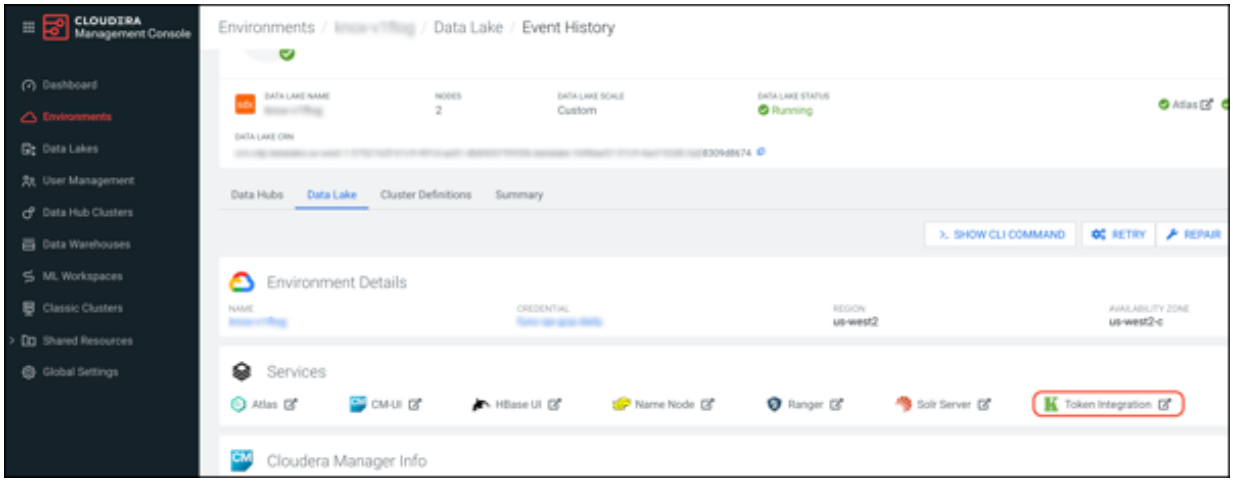
## View Knox token integration

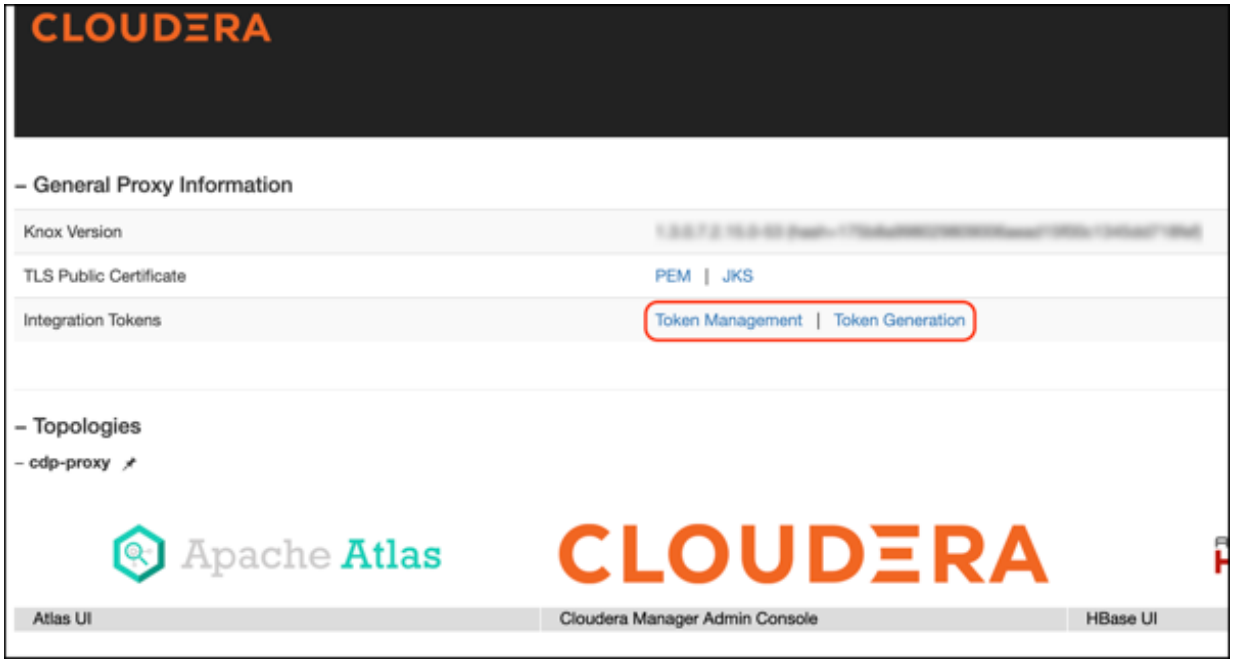
Knox token integration can be accessed via Cloudera Manager or the Knox homepage:

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



- 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 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.

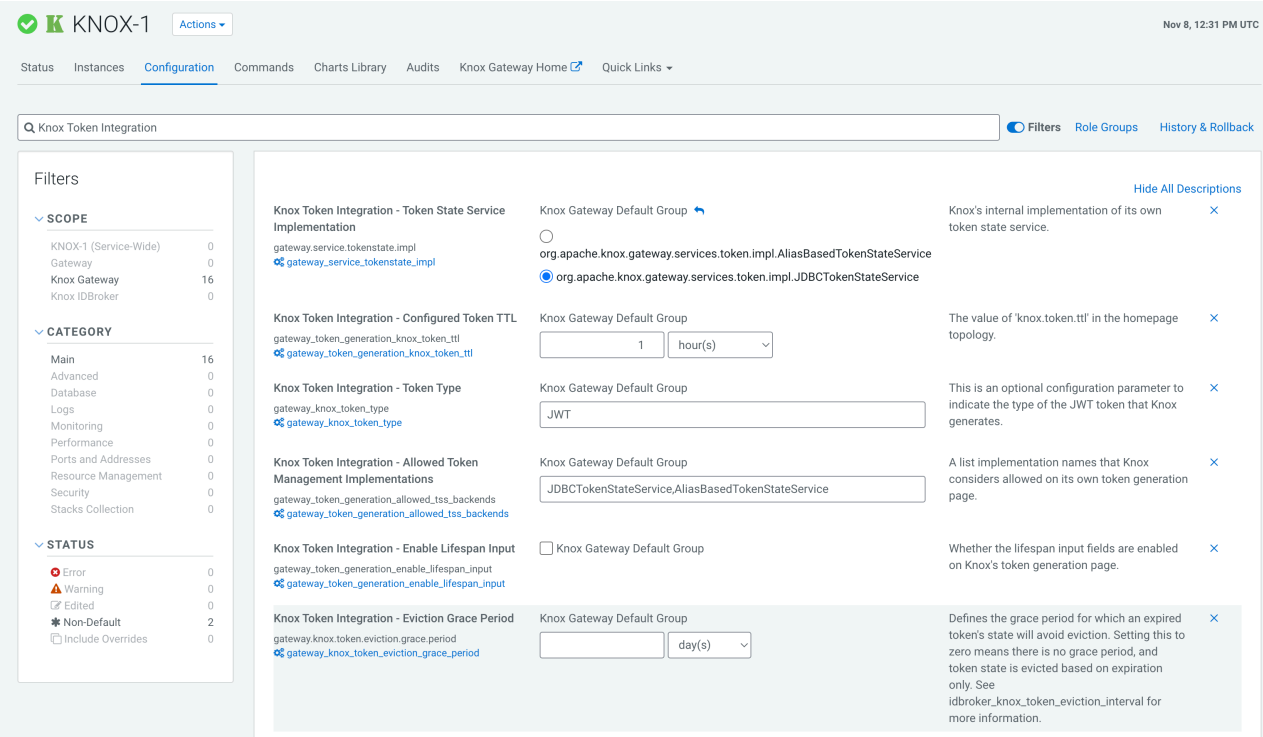
## Default configurations

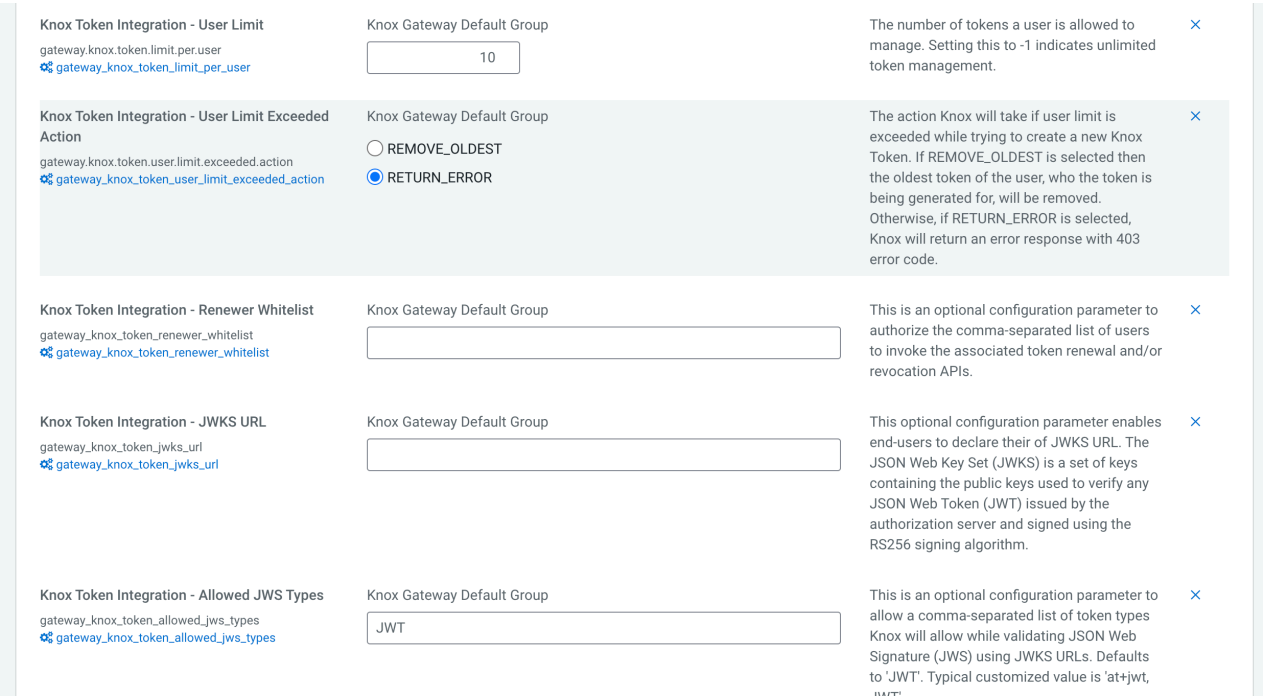
### Table 6: Default token configurations

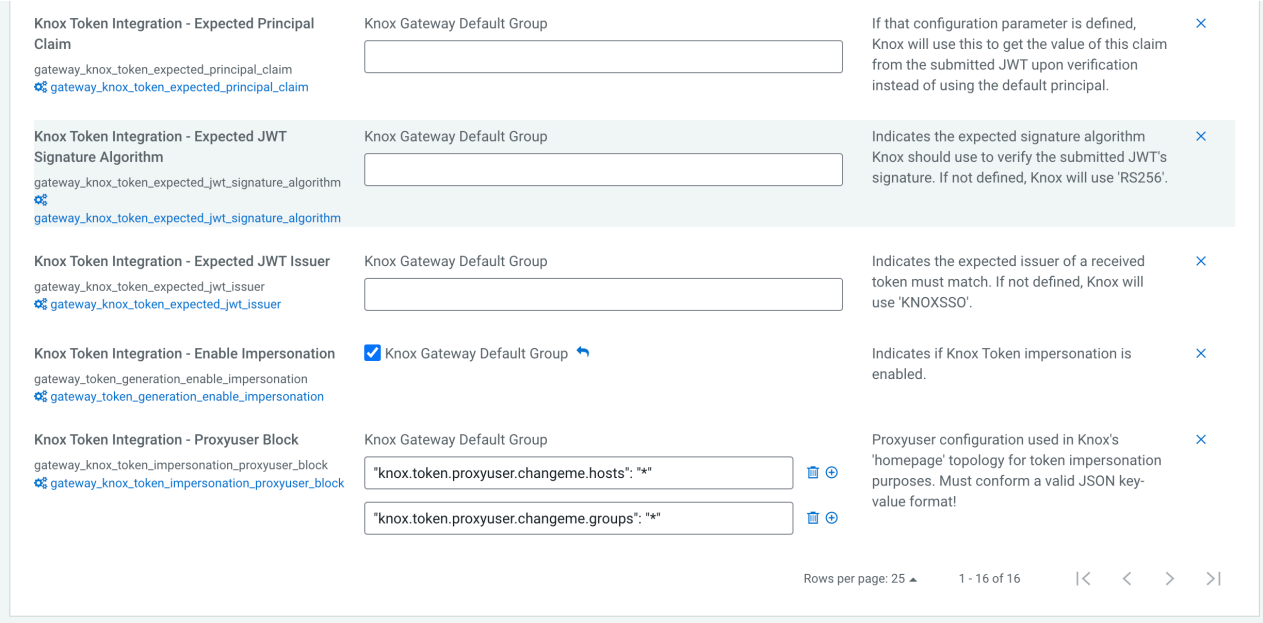
| Property | Sample values | Default |
|---|---|---|
| Token State Service Implementation | Knox's internal implementation of its own token state service. | org.apache.knox.gateway.services.token.impl.JDBCTokenStateService |
| Allowed Token Management Implementations | A list of implementation names 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 | Proxyuser configuration used in Knox's 'homepage' topology for token impersonation purposes. Must conform a valid JSON key-value format! | "knox.token.proxyuser.changeme.hosts": "*"<br>"knox.token.proxyuser.changeme.groups": "*" |

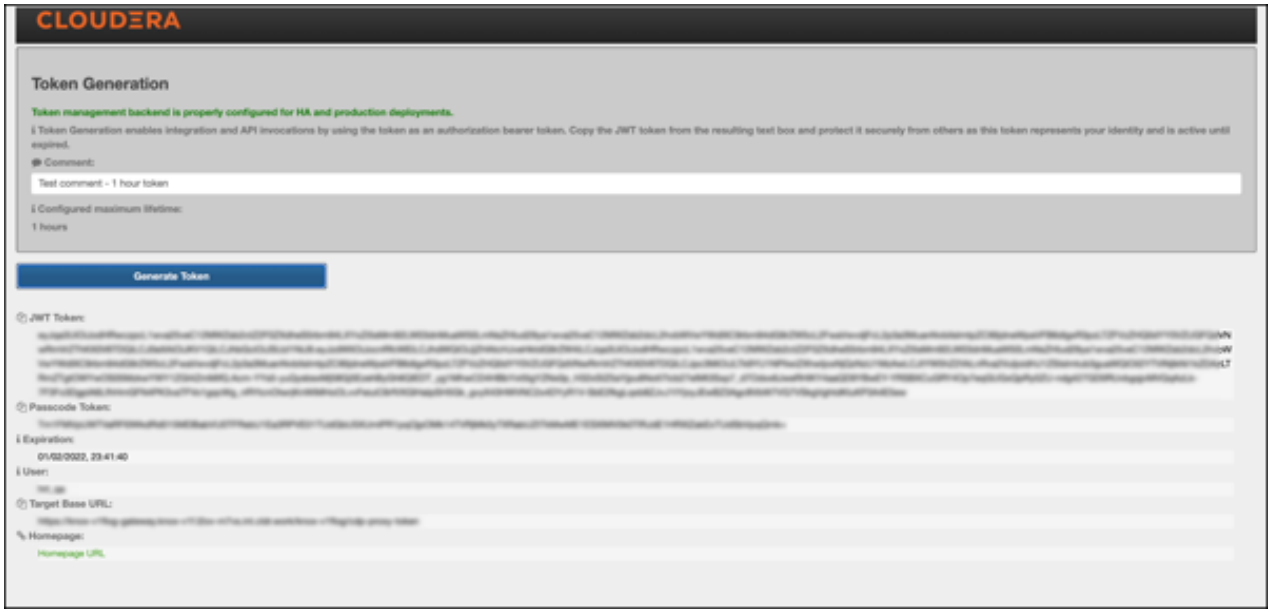
Default configurations seen from Cloudera Manager:






Default configurations seen from the Knox homepage UI:




## Database connection properties

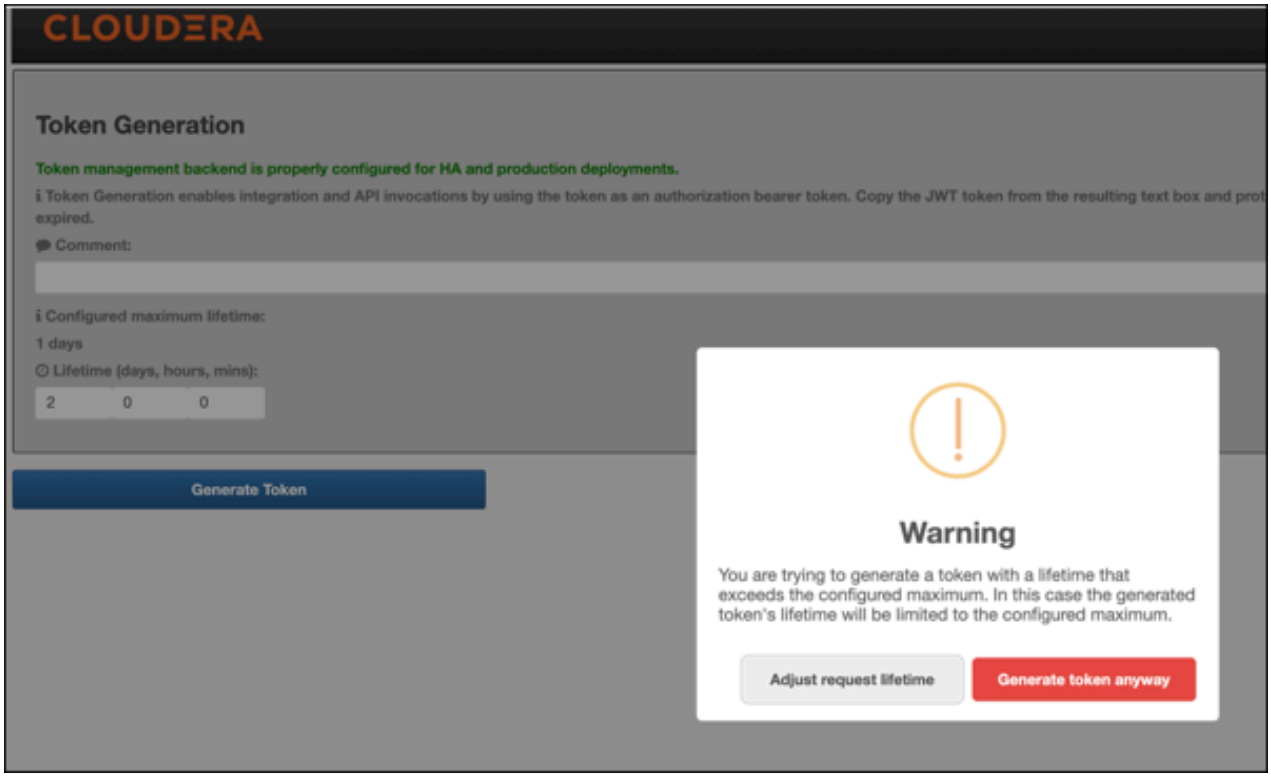
Optional database connection properties that you can declare individually:

- 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.

## 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 7: Options

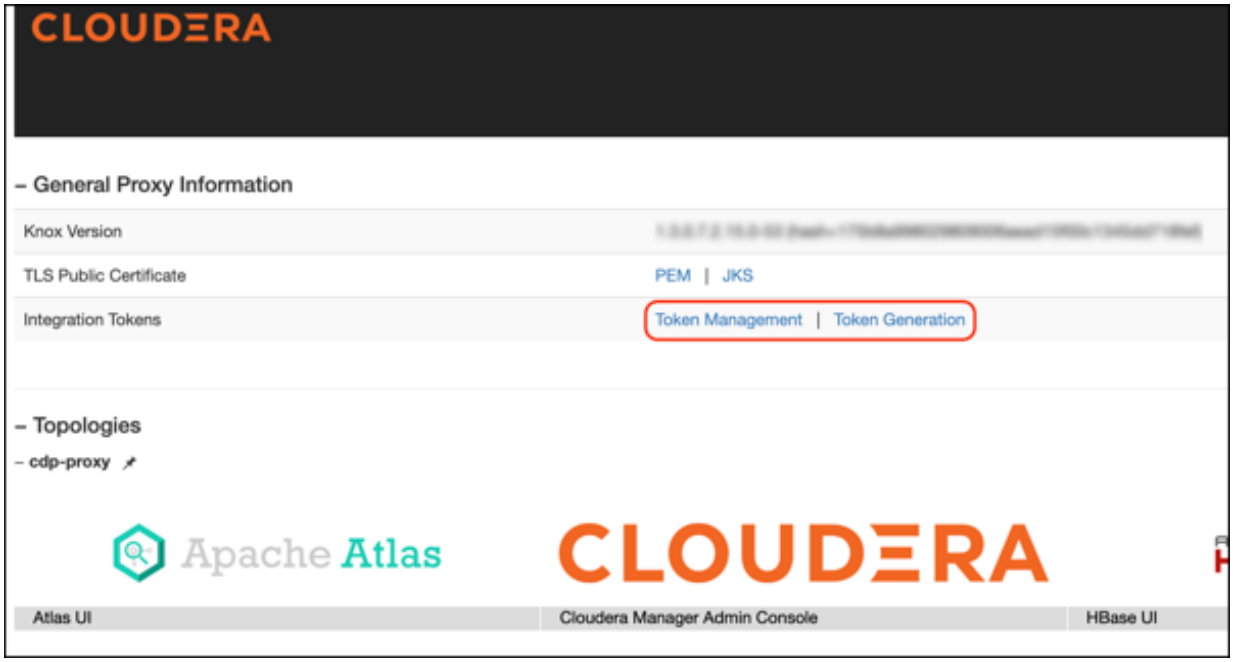
| Option | Description | Sample values |
|---|---|---|
| jwkAlg | (Optional) The desired JSON Web Signature algorithm name. Determines if the gateway-level alias is configured with a 256, 384, or 512-bit length JWK. | HS256 (Default)<br>HS384<br>HS512 |
| saveAlias | (Optional, Recommended) Given alias name used to save the generated JWK, instead of printing this sensitive information on the screen. | knox.token.hash.key |
| topology | (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. | cdp-proxy (Default)<br>cdp-proxy api |

## Generate tokens

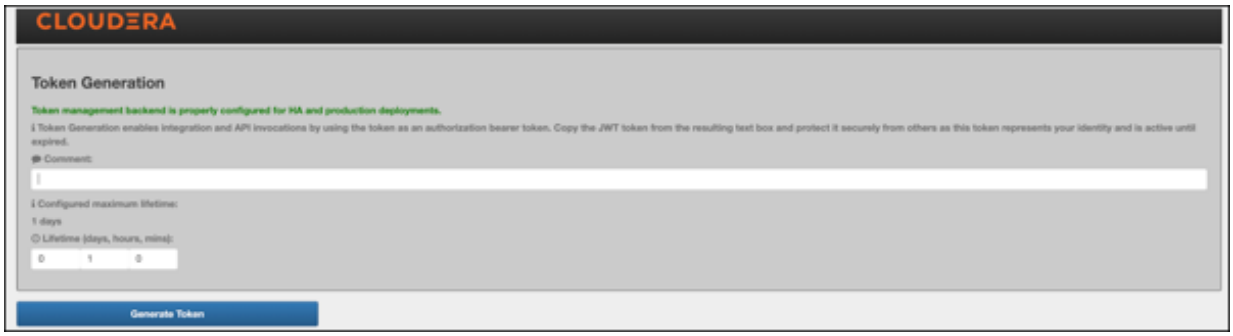
How to generate Knox gateway tokens from the Knox homepage.

### 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.



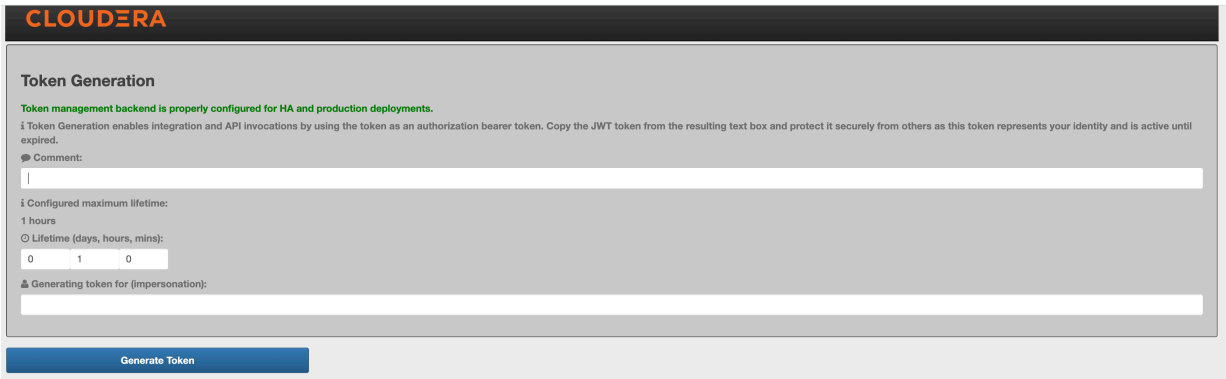
2. The following sections are displayed on the page:
   - Status bar: Message about the configured token state backend. There are 3 different statuses:
     - ERROR: Displayed in red. Indicates a problem with the service backend which makes the feature not work. Usually, this is visible when end-users 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 he 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.




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

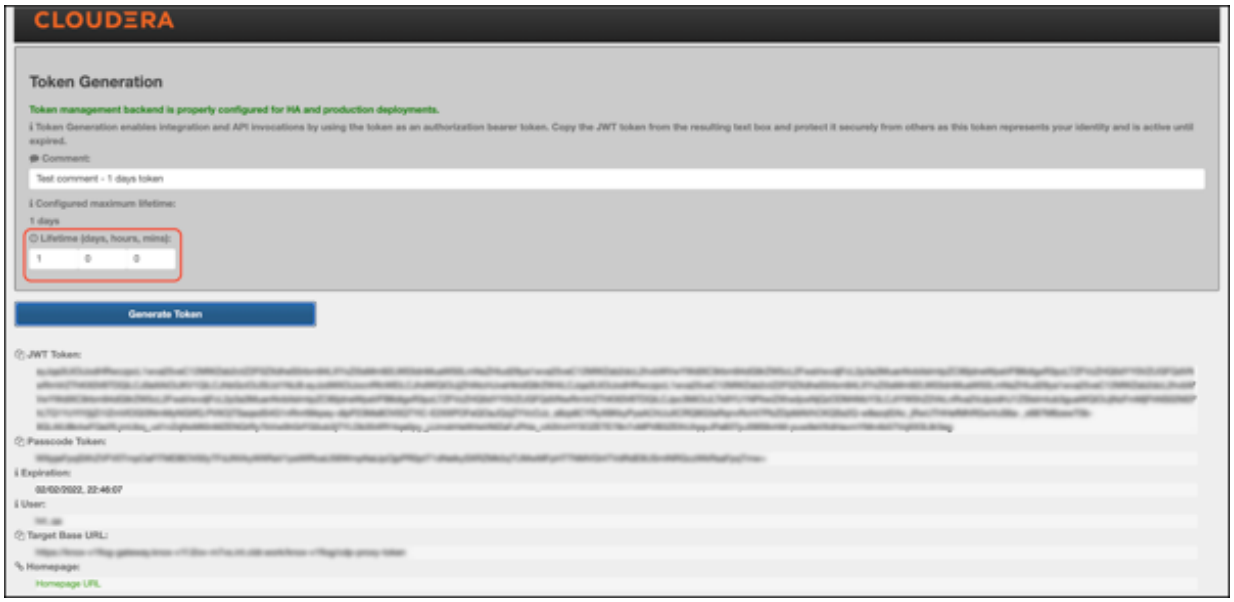
Using that input field our customers should be able to generate tokens on behalf of other users. For this to work, the Knox Token Integration - Proxyuser Block property has to be configured properly.

**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.




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:

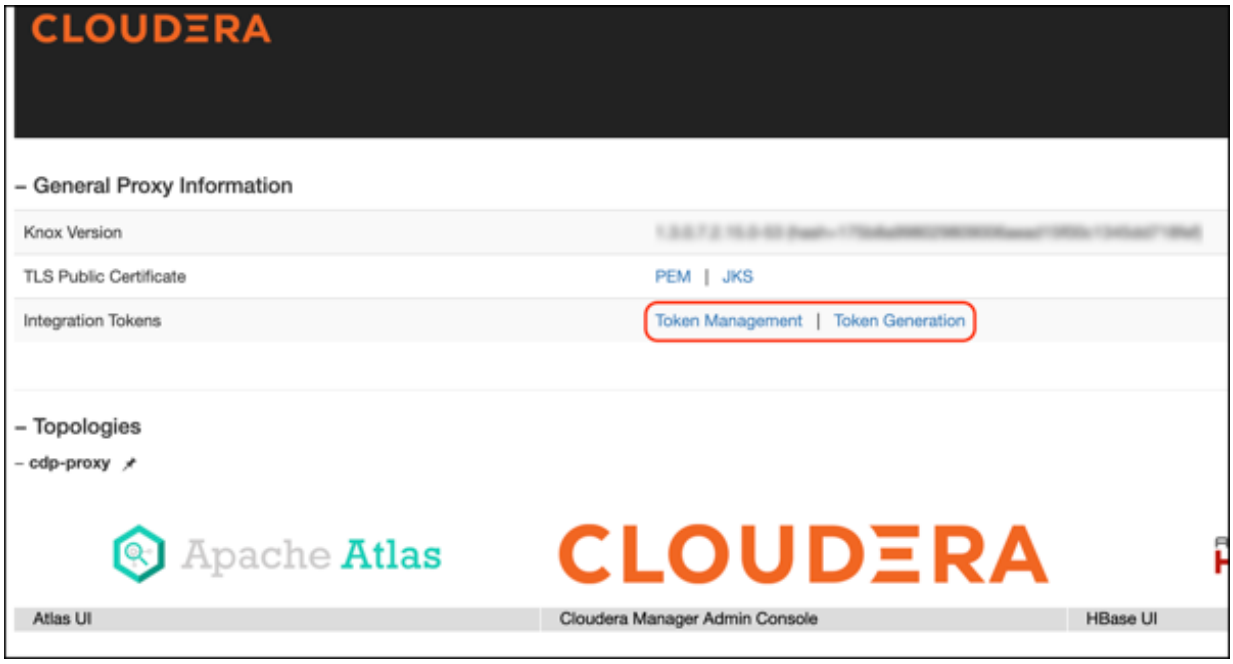
```
$ curl -ku Passcode:WkRFMk1XTmh[...]RVNFpXRTA= https://localhost:8443/ga
teway/cdp-proxy-token/webhdfs/v1?op=LISTSTATUS

{"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

1. 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.

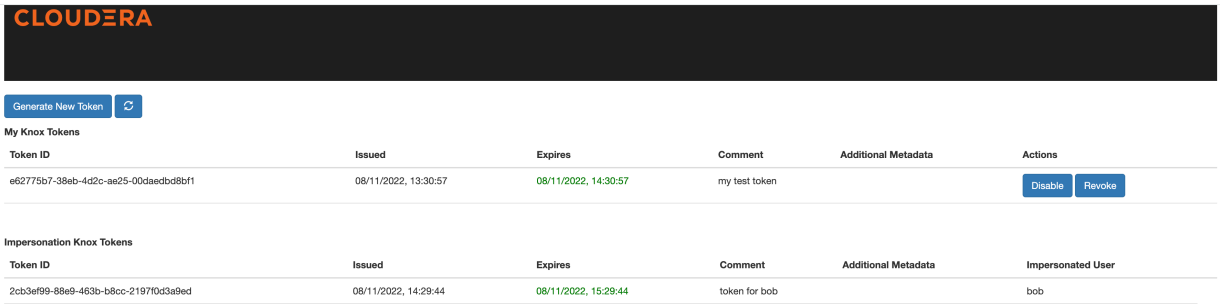



Active token will be displayed in green; expired tokens are red.

If Knox Token Integration - Enable Impersonation is set to true, the logged user will see two tables:

**a.** Tokens of the logged-in user
**b.** Tokens the logged-in user has generated for other users (impersonation)

The tables also display information about additional metadata




2. On this page, you will see basic information about your generated token(s) and you can execute the following actions:

   - Enable/Disable: Temporarily enable/disable a token.

     **Note:** Disabled tokens are not allowed to be use for authentication purposes.

   - Revoke: Permanently remove the token from the persistent store.

     **Caution:** This action cannot be undone; once you revoke a token, Knox will delete it from the in-memory cache and the underlying persistent token storage.

3. Click the Refresh icon above the table.

## 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/gatew
ay/sandbox/knoxtoken/api/v1/token?md_notebookName=accountantKnoxToken&md_sou
ldBeRemovedBy=31March2022&md_otherMeaningfuMetadata=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/gatew
ay/sandbox/knoxtoken/api/v1/token/getUserTokens?userName=admin&md_notebookNa
me=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.

# 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:** Conccurent 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:

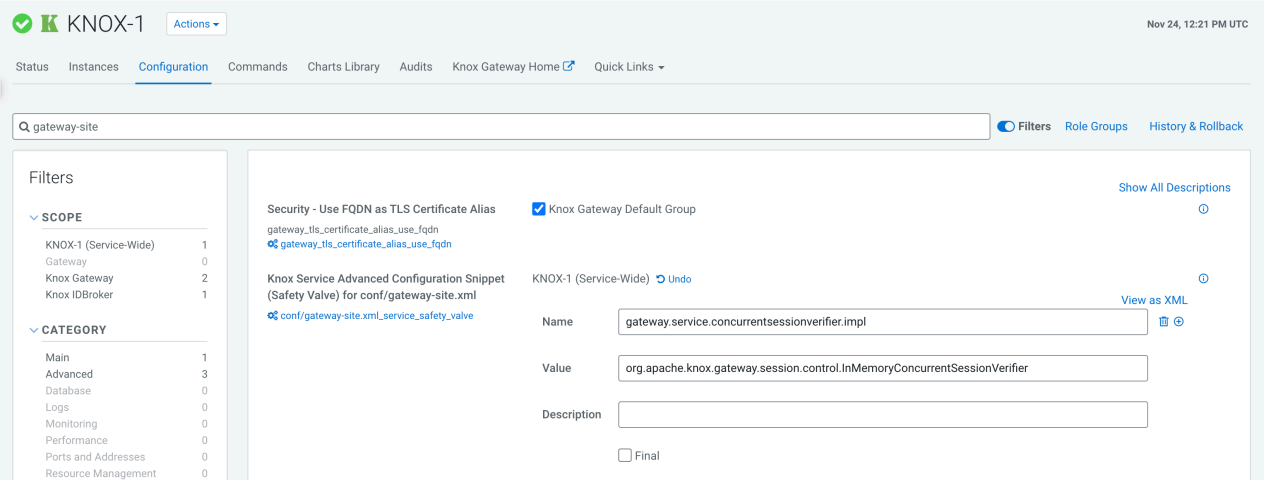
| Parameter | Description | Default |
|---|---|---|
| gateway.service.concurrentsessionverifier.impl | To enable the session verification feature, end-users should set this parameter to org.apache.knox.gateway.session.control.InMemoryConcurrentSessionVerifier | org.apache.knox.gateway.session.contr onVerifier |
| 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
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
Show All Descriptions
Security - Use FQDN as TLS Certificate Alias
gateway_tls_certificate_alias_use_fqdn
gateway_tls_certificate_alias_use_fqdn
Knox Gateway Default Group
Knox Service Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml
conf/gateway-site.xml_service_safety_valve
KNOX-1 (Service-Wide)
Undo
View as XML
Name
gateway.service.concurrentsessionverifier.impl
Value
org.apache.knox.gateway.session.control.InMemoryConcurrentSessionVerifier
Description
Final