

Cloudera Runtime 7.3.1

Using Cloudera Data Sharing

Date published: 2020-07-28

Date modified: 2024-12-10

CLOUDERA

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

Legal Notice

© Cloudera Inc. 2025. 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

How Cloudera Data Sharing works.....	4
Enabling Cloudera Data Sharing.....	4
Configuring Hive Metastore as a REST Catalog.....	6
Declaring Knox topologies.....	7
Knox configuration in gateway-site.xml.....	8
Configuring the Knox IDBroker.....	9
Creating a Data Share.....	11
Registering external clients in Cloudera.....	11
Managing Ranger policies.....	14
Importing the Cloudera certificate in the Spark cluster.....	16
Supported REST Catalog APIs for accessing the data.....	16

How Cloudera Data Sharing works

Cloudera Data Sharing enables your clients to run their workloads from data platforms, such as Databricks or Snowflake to fetch data from Cloudera environments for analytical purposes.

Cloudera Data Sharing involves the creation of a Data Share with the necessary authorization and authentication mechanisms. A Data Share is an organizational unit of data, a collection of data assets. You can then share this data with your clients so that they can access the Iceberg table data created within the Cloudera environment.

The following sections describe the high-level workflow of the processes involved in Cloudera Data Sharing.

Data Share creation

The following tasks are part of the Data Share creation process:

- As a resource owner, use the existing Knox Token Management system to generate a token. This unique Token ID is referred to as the `CLIENT_ID` and the generated passcode is the `CLIENT_SECRET`.
- As part of the token generation process, a Ranger role and Ranger group are created. This group is a virtual group that Knox provides for the client with whom the data is shared.
- Create and maintain policies for the set of databases and tables to be shared for the Ranger role and group and thereby create a Data Share.
- Maintain the `SELECT` permission for the databases or tables to allow `READ`-only access.
- You can then share the `CLIENT_ID` and the `CLIENT_SECRET` with your clients so that they can access the shared data in the Cloudera environment.

Data Share access

After you have created a Data Share, which includes creating tokens, authoring a read-only policy within Ranger, and shared the `CLIENT_ID` and `CLIENT_SECRET` with your client, the client makes use of these credentials in their workloads to establish a handshake with Cloudera and to access the shared data.

Related Information

[Enabling Cloudera Data Sharing](#)

[Creating a Data Share](#)

Enabling Cloudera Data Sharing

Learn how to perform the preparatory configurations in Cloudera to enable data sharing. These configurations are required for the creation of a data share in Cloudera and allow your clients to access data in Cloudera environments.

Installation and Upgrade Scenarios

Different version of Cloudera Runtime and Data Lakes require different steps to enable Cloudera Data Sharing.

Fresh 7.3.1.400 Data Lake Installation

Complete the following steps after a fresh installation of 7.3.1.400 Data Lake to enable .

1. [HMS Rest Catalog configuration](#)

Configure the Hive Metastore (HMS) service to serve as an Iceberg REST catalog. This allows clients to use REST Catalog APIs to access the required metadata files.

2. [Declaring Knox topologies](#)

Create the Knox topologies to define for Knox how to proxy requests from external users.

3. [Knox configuration](#)

Configure properties in Knox to set up administrator privileges that allow creation of users and tokens.

4. [IDBroker configuration](#)

Configure Knox IDBroker in Cloudera Manager.

5. [Creating a Data Share](#)

Create a Data Share in Cloudera

Upgrade from Cloudera Runtime 7.2.18.x with no REST Catalog / Cloudera Data Sharing configurations

Complete the following configurations to enable Cloudera Data Sharing when upgrade of Data Lake is done from 7.2.18.x with no REST Catalog or Cloudera Data Sharing configured in that version.

1. [Manual Installation of metering service](#)

Install and configure the metering server for Cloudera Data Sharing.

2. [HMS Rest Catalog configuration](#)

Configure the Hive Metastore service to serve as an Iceberg REST catalog. This allows your clients to use REST Catalog APIs to access the required metadata files.

3. [Declaring Knox topologies](#)

Create the Knox topologies to define for Knox how to proxy requests from external users.

4. [Knox configuration](#)

Configure properties in Knox to set up administrator privileges that allow creation of users and tokens.

5. [IDBroker configuration](#)

Configure Knox IDBroker in Cloudera Manager.

6. [Creating a Data Share](#)

Creating Data Share in Cloudera

Upgrade from Cloudera Runtime 7.2.18.x with REST Catalog / Cloudera Manager configuration

Following configurations are a must to enable Data Sharing when upgrade of Data Lake is done from 7.2.18.x with REST Catalog / DataSharing configured in that version.

1. [Manual Installation of metering service](#)

Install and configure the metering server for Cloudera Data Sharing.

2. [HMS Rest Catalog configuration](#)

Configure the Hive Metastore service to serve as an Iceberg REST catalog. This allows your clients to use REST Catalog APIs to access the required metadata files.

3. [Declaring Knox topologies](#)

Create the Knox topologies to define for Knox how to proxy requests from external users.

4. [Knox configuration](#)

Configure properties in Knox to set up administrator privileges that allow creation of users and tokens.

5. [IDBroker configuration](#)

Configure Knox IDBroker in Cloudera Manager.

6. [Creating a Data Share](#)

Creating Data Share in Cloudera

Resizing Cloudera Data Lake from Light Duty Data Lake to Enterprise Data Lake

Complete the following configurations are a must to enable Cloudera Data Sharing after resizing the Data Lake from Light Duty Data Lake to Enterprise Data Lake

1. HMS Rest Catalog configuration

Configure the Hive Metastore service to serve as an Iceberg REST catalog. This allows your clients to use REST Catalog APIs to access the required metadata files.

2. Declaring Knox topologies

Create the Knox topologies to define for Knox how to proxy requests from external users.

3. Knox configuration

Configure properties in Knox to set up administrator privileges that allow creation of users and tokens.

4. IDBroker configuration

Configure Knox IDBroker in Cloudera Manager.

Configuring Hive Metastore as a REST Catalog

To use the API endpoints provided by REST Catalog, you need to enable it in the Hive Metastore where it is deployed.

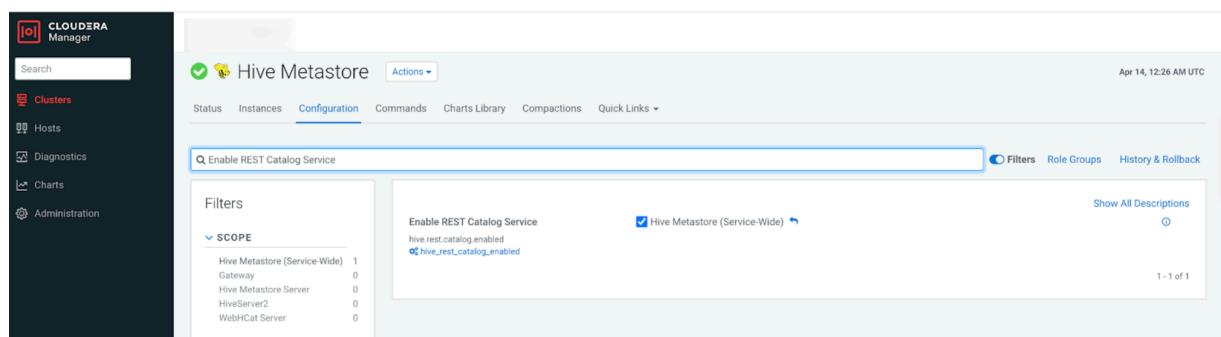
Before you begin

Make sure you have the IDBroker Host name before starting the steps.

To find the IDBroker Host, go to Cloudera Manager Knox Instances , and copy the hostname for the **Role** named **Knox IDBroker**.

Procedure

1. Log in to Cloudera Manager and click Clusters Hive Metastore Configuration .
2. Search for **Enable REST Catalog Service** and enable it to set up the REST Catalog Service in Hive Metastore.



Note: Use only one IDBroker, as Hive Metastore does not support multiple IDBrokers.

3. If you use Cloudera Data Sharing with AWS Elastic MapReduce and AWS Athena notebook, add the region of AWS environment to the Hive Metastore configuration.
 - a) Check your region in Cloudera Management Console Environments [***YOUR_ENVIRONMENT***] Environment Details .
 - b) Go to Cloudera Manager Clusters Hive Metastore Configuration Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml .
 - c) Add a property with your region.


```
<property><name>client.region</name><value>[***CLIENT_REGION_NAME***]</value></property>
```
4. Save the changes and restart the Hive Metastore service.

What to do next

Continue with [declaring the Knox topologies](#).

Declaring Knox topologies

Learn how to create the Knox topologies to define for Knox how to proxy requests from external users.

Before you begin

Ensure that you have the following information before performing the steps:

- HMS Host and Ranger Host: To get the HMS Host and Ranger Admin Host, go to Data Lake Nodes , and copy the **FQDN** of the Master node.
- Username of the user who will run the script to generate the CLIENT_ID and CLIENT_SECRET.

Procedure

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

```
NAME:
providerConfigs:cdp-share-access-providers
VALUE:
role=federation#federation.name=JWTProvider#federation.enabled=true#federation.param.knox.token.exp.server-managed=true#role=identity-assertion#identity-assertion.name=Default#identity-assertion.enabled=true#identity-assertion.param.group.mapping.$PRIMARY_GROUP=(not (member username))

NAME:
cdp-share-access
VALUE:
providerConfigRef=cdp-share-access-providers#KNOXTOKEN:knox.token.ttl=36000000#KNOXTOKEN:knox.token.exp.server-managed=true#KNOXTOKEN:gateway.knox.token.limit.per.user=-1#HMS-API:url=http://[***HMS-HOST***]:8090

NAME:
providerConfigs:cdp-share-management-providers
VALUE:
role=authentication#authentication.name=ShiroProvider#authentication.param.main.invalidRequest=org.apache.shiro.web.filter.InvalidRequestFilter#authentication.param.main.invalidRequest.blockBackslash=false#authentication.param.main.invalidRequest.blockNonAscii=false#authentication.param.main.invalidRequest.blockSemicolon=false#authentication.param.main.pamRealm=org.apache.knox.gateway.shirorealm.KnoxPamRealm#authentication.param.main.knoxAnonFilter=org.apache.knox.gateway.filter.AnonymousAuthFilter#authentication.param.urls./knoxtoken/api/v1/jwks.json=knoxAnonFilter#authentication.param.main.pamRealm.service=login#authentication.param.sessionTimeout=30#authentication.param.urls./**=authcBasic#role=identity-assertion#identity-assertion.name=HadoopGroupProvider#identity-assertion.param.hadoop.proxyuser.impersonation.enabled=true#identity-assertion.param.hadoop.proxyuser.[***USER-WHO-RUNS-THE-SCRIPT]***.users=*#identity-assertion.param.hadoop.proxyuser.[***USER-WHO-RUNS-THE-SCRIPT]***.groups=*#identity-assertion.param.hadoop.proxyuser.[***USER-WHO-RUNS-THE-SCRIPT]***.hosts=*#identity-assertion.param.CENTRAL_GROUP_CONFIG_PREFIX=gateway.group.config.#role=authorization#authorization.name=XASecurePDPKnox#authorization.enabled=false#role=ha#ha.name=HaProvider#ha.enabled=true
```

```
#ha.param.RANGER=enableStickySession=false;noFallback=false;enableLoadBalancing=true
```



Note: The default value of 36000000 of `knox.token.ttl` means a 10 hour period. You can reduce this number to adjust the `CLIENT_ID` and `CLIENT_SECRET` lifetime.



Note: If additional users need to run the script, the following section needs to be duplicated to add the user.

```
#identity-assertion.param.hadoop.proxyuser.[***USER-WHO-RUNS-THE-SCRIPT]***.users=#identity-assertion.param.hadoop.proxyuser.[***USER-WHO-RUNS-THE-SCRIPT]***.groups=#identity-assertion.param.hadoop.proxyuser.[***USER-WHO-RUNS-THE-SCRIPT]***.hosts=*
```

```
NAME:
cdp-share-management
VALUE:
providerConfigRef=cdp-share-management-providers#RANGER:url=http
ps://[***RANGER-HOST***]:6182#KNOXTOKEN:knox.token.ttl=-1#KNOXTOKEN:knox
.token.type=JWT#KNOXTOKEN:knox.token.target.url=cdp-proxy-token#KNOXTOKEN:knox.token.audiences=cdp-proxy-token#KNOXTOKEN:knox.token.client.data=
homepage_url=homepage/home?profile=token&topologies=cdp-proxy-token#KNOXTOKEN:knox.token.exp.tokengen.allowed.tss.backends=JDBCTokenStateService,AliasBasedTokenStateService#KNOXTOKEN:knox.token.lifespan.input.enabled=true#KNOXTOKEN:knox.token.user.limit.exceeded.action=RETURN_ERROR#KNOXTOKEN:knox.token.exp.server-managed=true
```

The relevant Knox topologies are created.

4. Click Save Changes.

What to do next

Continue with [configuring Knox](#).

Related Information

[Knox topologies](#)

Knox configuration in gateway-site.xml

Learn how to configure Knox parameters to allow admin permissions in Knox. Admin permission is required to create the `CLIENT_ID` and `CLIENT_SECRET`.

About this task

The `CLIENT_ID` and `CLIENT_SECRET` is required for creating Data Shares to authorize your external clients.

Before you begin

- The Cloudera public cloud user must be configured as both Knox and Ranger Admin to perform the tasks required to configure Knox parameters.
- Declare [the Knox topologies](#).

Procedure

1. Go to Cloudera Manager Knox Instances Configuration Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml

2. Add the gateway.knox.admin.users parameter.

**Note:**

Users who run the knoxshare.py script, must be a part of the Knox admin users and groups configuration. For example, gateway.knox.admin.users = shareadmin1, shareadmin2.

3. Add the gateway.knox.admin.groups parameter.
4. Add the gateway.knox.token.limit.per.user parameter.



Note: The value -1 means "unlimited". For example, using gateway.knox.token.limit.per.user=2 allows two Data Share users.

The screenshot shows the Cloudera Manager interface for configuring the Knox Gateway. The left sidebar contains navigation links: Clusters, Hosts, Diagnostics, Charts, and Administration. The main panel is titled 'Knox' and includes tabs for Status, Instances, Configuration, Commands, Charts Library, and Quick Links. The 'Configuration' tab is active, showing a search bar and filters. The configuration settings are organized into sections: Filters (SCOPE and CATEGORY), Auto Discovery - Advanced Configuration (Monitoring Interval), Knox Gateway JDBC URL Override, System User, System Group, and Knox Service Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml. The 'Knox Service Advanced Configuration Snippet' section shows a configuration snippet for gateway-site.xml with properties for admin users, admin groups, and token limit per user.

What to do next

Continue with [configuring the Knox IDBroker](#).

Related Information

[Declaring Knox topologies](#)

Configuring the Knox IDBroker

Learn how to configure the Knox IDBroker in Cloudera Manager.

About this task

The IDBroker must be made aware of available session policies. Configure these policies using the Cloudera Manager such so that they survive restarts, upgrades, and other such events.

Procedure

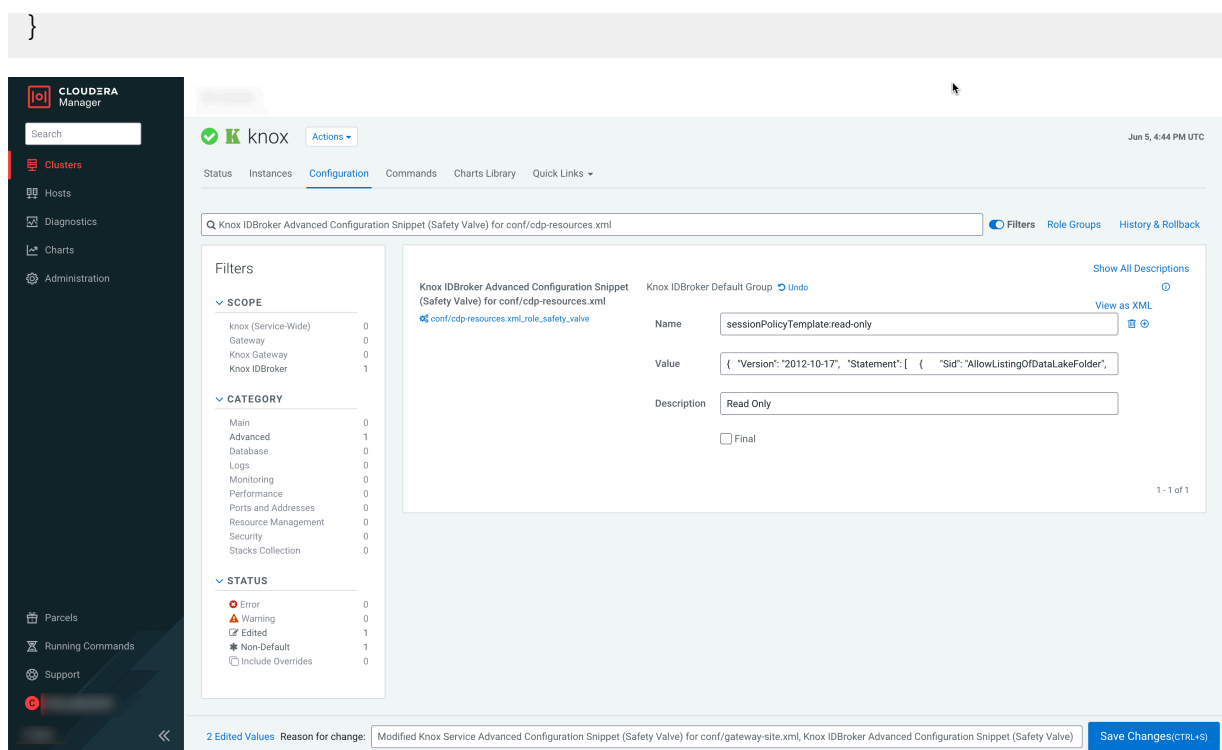
1. Go to Cloudera Manager Knox Instances Configuration Knox IDBroker Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml
2. Add a property named sessionPolicyTemplate:read-only with the following values:

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "AllowListingOfDataLakeFolder",
    "Effect": "Allow",
    "Action": [
      "s3:GetAccelerateConfiguration",
      "s3:GetAnalyticsConfiguration",
      "s3:GetBucketAcl",
      "s3:GetBucketCORS",
      "s3:GetBucketLocation",
      "s3:GetBucketLogging",
      "s3:GetBucketNotification",
      "s3:GetBucketPolicy",
      "s3:GetBucketPolicyStatus",
      "s3:GetBucketPublicAccessBlock",
      "s3:GetBucketRequestPayment",
      "s3:GetBucketTagging",
      "s3:GetBucketVersioning",
      "s3:GetBucketWebsite",
      "s3:GetEncryptionConfiguration",
      "s3:GetInventoryConfiguration",
      "s3:GetLifecycleConfiguration",
      "s3:GetMetricsConfiguration",
      "s3:GetObject",
      "s3:GetObjectAcl",
      "s3:GetObjectTagging",
      "s3:GetObjectVersion",
      "s3:GetObjectVersionAcl",
      "s3:GetObjectVersionTagging",
      "s3:GetReplicationConfiguration",
      "s3:ListBucket",
      "s3:ListBucketMultipartUploads",
      "s3:ListMultipartUploadParts"
    ],
    "Resource": "arn:aws:s3:::${bucket}",
    "Condition": {
      "StringEquals": {
        "s3:prefix": [
          "${prefix}",
          "${prefix}/*"
        ]
      }
    }
  }
]

```



3. Save your changes and restart the Hive Metastore service.



Note:

If the IDBroker service indicates that it has stale a configuration, restart it.

What to do next

Continue with [creating a Data Share](#).

Creating a Data Share

Learn how resource owners or Data Share administrators can share Iceberg tables in Cloudera by registering external clients in Cloudera Cloud and configuring Ranger policies.

Resource owners or Data Share administrators who want to share their Iceberg tables in Cloudera with external clients must first register the client in the Cloudera on cloud environment. After that, the resource owner needs to configure Ranger policies to allow access for the external client.

Registering external clients in Cloudera

Learn how to register external clients in Cloudera to provision a `CLIENT_ID` and `CLIENT_SECRET`.

Before you begin



Note: Users who run the APIs, must be a part of the Knox admin users and groups configuration. For more information see [Knox configuration in gateway-site.xml](#).

Ensure that you have the following information before performing the steps:

Share Admin user and password

Username and password of the Cloudera Administrator

Knox hostname

To get the Knox hostname, go to Cloudera Manager Knox Instances , and copy the hostname for the **Knox Gateway** role.

Data Lake name

Go to Management Console Environments <***YOUR_ENVIRONMENT_NAME***> Data Lake Details and copy and make a note of the Data Lake name.

Procedure

1. Create the CLIENT_ID and SECRET in Knox by running the following command:

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***] https://[***KNOX-HOST-NAME***]:8443/[***DATA LAKE NAME***]/cdp-share-management/knoxtoken/api/v1/token?doAs=external.user&comment=[***<ADDITION INFO>***]&md_contact=[***EMAIL_ID***]&md_role=[***ROLE OF THE CLIENT***]&md_type=[***CLIENT_ID***]" | jq -r '"CLIENT_ID: \(.token_id) SECRET: \(.passcode)'"
```

- doAs=external.user - Set value to external.user
- comment - Additional comments on the CLIENT
- md_contact - Client contact metadata, for example, email_id
- md_role - <[***ROLE FOR THE CLIENT_ID***]>
- md_type - Set value to CLIENT_ID

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***] "https://my-datalake-name.int.cldr.work:8443/my-datalake-name/cdp-share-management/knoxtoken/api/v1/token?doAs=external.user&comment=carriers&md_contact=client_name@company.com&md_role=UnitedAirlinesRole&md_type=[***CLIENT_ID***]" | jq -r '"CLIENT_ID: \(.token_id) SECRET: \(.passcode)'"
CLIENT_ID : 462babd3-fe5a-4abf-8b47-526897677ad5 SECRET: TkRZeVltRmlaRE10Wm1VMV1TMDBV0ptTFRoaU5EY3ROVEkyT0RrM05qYzNZV1ExOjpaalV4WWpReFkyWXRObV V6WlMwME4yTm1MVGcyWWpFdE9XSXhZekE0TlRZMU9XTmw= '
```

By Enterprise Data Lakes, use the following command:

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***] https://[***LOAD BALANCER***]/[***DATA LAKE NAME***]/cdp-share-management/knoxtoken/api/v1/token?doAs=external.user&comment=[***<ADDITION INFO>***]&md_contact=[***EMAIL_ID***]&md_role=[***ROLE OF THE CLIENT***]&md_type=[***CLIENT_ID***]" | jq -r '"CLIENT_ID: \(.token_id) SECRET: \(.passcode)'"
```

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***]:"https://my-loadbalancer-1745504451479-b5765eae5b22d08.elb.us-west-2.amazonaws.com/dldamedi-28qgc9/cdp-share-management/knoxtoken/api/v1/token?doAs=external.user&comment=carriers&md_contact=client_name@company.com&md_role=UnitedAirlinesRole&md_type=[***CLIENT_ID***]" | jq -r '"CLIENT_ID: \(.token_id) SECRET: \(.passcode)'"
```



Note:

To verify if the CLIENT_ID (Token ID) is generated successfully, go to Cloudera Management Console Environment Data Lake Token Integration Token Management .

2. Create a Ranger Group with Client ID as the name of that Group with the following command:

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***] -H "Accept: application/json" -H "Content-Type: application/json" -X POST "https://[***RANGER-HOST-NAME***]:8443/[***DATA LAKE NAME***]/cdp-share-manag
```

```
ement/ranger/service/xusers/groups/" -d '{"name": "[***CLIENT_ID***]", "description": "group representing a share for a CLIENT_ID"}'
```

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***] -H "Accept: application/json" -H "Content-Type: application/json" -X POST "https://my-datalake.int.clldr.work:8443/my-datalake-name/cdp-share-management/ranger/service/xusers/groups/" -d '{"name": "462babd3-fe5a-4abf-8b47-526897677ad5", "description": "group representing a share for a CLIENT_ID"}'
```



Note: Apache Ranger provides the following public API to create the Group. For more information, see the [Apache Ranger API documentation \(POST /xusers/groups\)](#).

By Enterprise Data Lakes, use the following command:

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***] -H "Accept: application/json" -H "Content-Type: application/json" -X POST "https://[***RANGER-HOST-NAME***]/[***DATA LAKE-NAME***]/cdp-share-management/ranger/service/xusers/groups/" -d '{"name": "[***CLIENT_ID***]", "description": "group representing a share for a CLIENT_ID"}'
```

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***] -H "Accept: application/json" -H "Content-Type: application/json" -X POST "https://my-gateway-hostname.clldr.work:8443/my-datalake-name/cdp-share-management/ranger/service/xusers/groups/" -d '{"name": "462babd3-fe5a-4abf-8b47-526897677ad5", "description": "group representing a share for a CLIENT_ID"}'
```

3. Create a new Role and add the created Group to the Role in Ranger with the following command

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***] -H "Accept: application/json" -H "Content-Type: application/json" -X POST "https://[***RANGER-HOST-NAME***]:8443/[***DATA LAKE-NAME***]/cdp-share-management/ranger/service/public/v2/api/roles/" -d '{"name": "[***CLIENT_ROLE***]", "description": "[***CLIENT_ROLE DESCRIPTION***]", "groups": [ { "name": "[***CLIENT_ID CREATED***]", "isAdmin": false } ] }'
```

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***] -H "Accept: application/json" -H "Content-Type: application/json" -X POST "https://my-ranger-hostname.int.clldr.work:8443/my-datalake-name/cdp-share-management/ranger/service/public/v2/api/roles/" -d '{"name": "SalesDeptRole", "description": "SalesRole description", "groups": [ { "name": "462babd3-fe5a-4abf-8b47-526897677ad5", "isAdmin": false } ] }'
```



Note: Apache Ranger provides the following public API to create the a Role. For more information, see the [Apache Ranger API documentation \(POST /roles/roles\)](#).

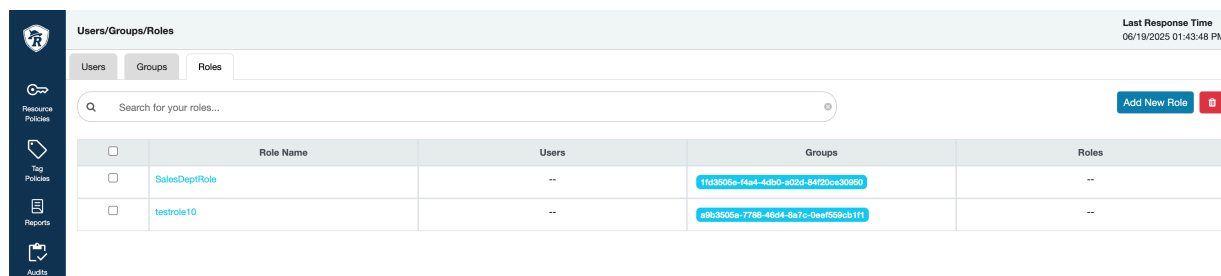
By Enterprise Data Lakes and, use the following command:

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***] -H "Accept: application/json" -H "Content-Type: application/json" -X POST "https://[***GATEWAY-HOST-NAME***]/[***DATA LAKE-NAME***]/cdp-share-management/ranger/service/public/v2/api/roles/" -d '{"name": "[***CLIENT_ROLE***]", "description": "[***CLIENT_ROLE DESCRIPTION***]", "groups": [ { "name": "[***CLIENT_ID CREATED***]", "isAdmin": false } ] }'
```

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***] -H "Accept: application/json" -H "Content-Type: application/json" -X POST "https://my-gateway-hostname.int.clldr.work:8443/my-datalake-name/cdp-share-management/ranger/service/public/v2/api/roles/" -d '{"name": "TestRole1", "description":
```

```
"TestRole1 description", "groups": [ { "name": "test-group1", "isAdmin": false } ] }'
```

Figure 1: Apache Ranger Settings Roles



	Role Name	Users	Groups	Roles
<input type="checkbox"/>	SalesDeptRole	--	1f6508e-14e4-4db0-a0d1-84f50ce30960	--
<input type="checkbox"/>	testrole10	--	a9b3505a-7788-46d4-8a7c-0eaf559cd3f1	--



Note:

The provisioned CLIENT_ID and CLIENT_SECRET are used by the external client to get authenticated into Cloudera as part of an OAuth client credentials flow.

4. Add the Group to an existing Role with the following commands:

- Get the RoleId for the RoleName from Ranger Admin via API:

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***] -H "Accept: application/json" -H "Content-Type: application/json" -X GET "https://[***RANGER-HOST-NAME***]:8443/[***DATA LAKE-NAME***]/cdp-share-management/ranger/service/public/v2/api/roles/name/<name>" | jq -r '"RoleId: \(.id) "'
```

- Add the Group to the Role using RoleId

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***] -H "Accept: application/json" -H "Content-Type: application/json" -X PUT "https://[***RANGER-HOST-NAME***]:8443/[***DATA LAKE-NAME***]/cdp-share-management/ranger/service/public/v2/api/roles/<RoleId>" -d '{ "name": "***CLIENT_ROLE***", "description": "***CLIENT_ROLE DESCRIPTION***", "groups": [ { "name": "***CLIENT_ID CREATED***", "isAdmin": false } ] }'
```

Results

The registration process results in provisioning a CLIENT_ID and CLIENT_SECRET followed by creating Ranger ROLE and adding CLIENT_ID as a Group to the ROLE and then maintaining policy for the ROLE to create the data share.

Managing Ranger policies

Learn how to provide authentication capabilities to your external users. Manage and govern your Ranger policies.

About this task

The Ranger Administrator must maintain policies for the set of databases and tables for the Ranger role and group.

Figure 2: Apache Ranger Resource Policies Hadoop SQL

Before you begin

In the Allow Conditions, a “SELECT” permission has to be maintained for the Databases or Tables to provide a READ-only access.

Figure 3: Apache Ranger Resource Policies Hadoop SQL Allow Conditions

Procedure

Create a Data Share policy in Ranger for the previously created Role with the following command

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***] -H "Accept: application/json" -H "Content-Type: application/json" -X POST "https://[***RANGER-HOST-NAME***]:8443/[***DATA LAKE-NAME***]/cdp-share-management/ranger/service/public/v2/api/policy/" -d '{"service": "hive_service_name", "policyType": 0, "name": "Iceberg Table Policy", "description": "Policy for SELECT access to an CLIENT_ID", "isEnabled": true, "resources": { "database": { "values": ["***DATABASE_NAME***"] }, "table": { "values": ["***TABLE_NAME***"] }, "column": { "values": ["*"] } }, "policyItems": [ { "accesses": [ { "type": "select" } ], "users": [], "groups": [], "roles": ["***CLIENT_ROLE***"], "conditions": [ ] } ] }'
```



Note: For more information on creating Ranger policies, see the [Apache Ranger API documentation \(POST /public/v2/api/policy\)](#).

```
curl -k -u [***CDP_ADMIN_USER***]:[***PASSWORD***] -H "Accept: application/json" -H "Content-Type: application/json" -X POST "https://dldanew-vxxt5w-master0.dldanew.svbr-nqvp.int.clldr.work:8443/dldanew-vxxt5w/cdp-share-management/ranger/service/public/v2/api/policy/" -d '{"service": "cm_hive", "policyType": 0, "name": "Hive Table Policy", "description": "Policy for SELECT access to an external user", "isEnabled": true, "resources": { "database": { "values": ["emp_data"] }, "table": { "values": ["employees"] }, "column": { "values": ["*"] } }, "policyItems": [ { "accesses": [ { "type": "select" } ], "users": [], "groups": [], "roles": ["testrole13"], "conditions": [ ] } ] }'
```

Related Information[Ranger Policies Overview](#)

Importing the Cloudera certificate in the Spark cluster

If your environment contains a custom set of trusted certificate authorities, you need to import the Cloudera certificate to your environment and apply it to the node where Spark is running.

Procedure

1. Run the following command to export a certificate from the Cloudera environment:

```
openssl s_client -showcerts -connect [***CDP-PUBLIC-CLOUD-HOSTNAME***]:443
</dev/null 2>/dev/null | openssl x509 -outform PEM > share.pem
```

2. Convert the created share.pem file to share.cert by running the following command:

```
openssl x509 -outform der -in share.pem -out share.cert
```

3. Import the certificate into the Java Virtual Machine (JVM) of the Spark cluster.

```
keytool -importcert -alias cdpcert -keystore /usr/lib/jvm/java-11-openjdk-
amd64/lib/security/cacerts -file ${CERT_HOME}/share.cert -storepass change
it -noprompt
```



Note: The JAVA path, /usr/lib/jvm/java-11-openjdk-arm64/lib/security/cacerts is different for ARM-based machines.

Supported REST Catalog APIs for accessing the data

To use the Iceberg data in Cloudera, you must employ the REST client to perform specific operations.

The REST APIs from the specification defined by Apache Iceberg are available in the [REST Catalog open API specification](#). Cloudera currently supports the REST APIs that allow read operations on Iceberg tables:



Note: The {prefix} part is optional. It can be replaced by a prefix to differentiate between API environments or instances. For more information, see the [Iceberg REST Catalog Open API specification](#).

Retrieving the access token

Retrieving the access token works the same for all supported endpoints:

```
[***MY-TOKEN-NAME***]=$(\curl -k -X POST -H "Content-Type: applic
ation/x-www-form-urlencoded" -d "client_id=[***CLIENT ID***]&cli
ent_secret=[***CLIENT SECRET***]&grant_type=client_credentials" "htt
ps://[***DATA LAKE-LOADBALANCER***]/[***DATA LAKE-NAME***]/cdp-share-access/hm
s-api/icecli/v1/oauth/tokens" | jq -r '.access_token')
```

Using the endpoints must be always preceded by retrieving the access token.

List Databases: /v1/{prefix}/namespaces

List all namespaces at a certain level, optionally starting from a given parent namespace.

```
curl -ivk -X GET -H "Content-Type: application/x-www-form-urlencoded" -H "Authorization: Bearer $[***MY-TOKEN-NAME***]" https://[***DATA LAKE-LOADBALANCER***]/[***DATA LAKE-NAME***]/cdp-share-access/hms-api/icecli/v1/namespaces
```



Note: The `[***DATA LAKE-LOADBALANCER***]` DNS can be found via Cloudera Management Console in the **Load Balancers** section.

Example:

```
My-token-name=$(curl -k -X POST -H "Content-Type: application/x-www-form-urlencoded" -d "client_id=b9efc3dd-3695-4867-9e4c-523389c8a78b&client_secret=WpsbFptTXpar1F0TXpZNU5TMDBPRFkzTFRsbE5HTXROVEL6TXpnNVl6aGhOemhpOjpNRGt5TmnpneElHUXRZak00TmkwMFpqTXhMVGczTTJZdFptTXhOekl6TmnpVNF1qazI=&grant_type=client_credentials" "https://apr24-LBInternal-1745472822408-fddbd702bfadbe45.elb.us-west-2.amazonaws.com/apr24-env2-dl/cdp-share-access/hms-api/icecli/v1/oauth/tokens" | jq -r '.access_token')
```

```
curl -ivk -X GET -H "Content-Type: application/x-www-form-urlencoded" -H "Authorization: Bearer $My-token-name" https://apr24-LB-my-datalake-load-balancer.elb.us-west-2.amazonaws.com/my-datalake-name/cdp-share-access/hms-api/icecli/v1/namespaces
```

List Tables: /v1/{prefix}/namespaces/{namespace}/tables

List all table identifiers under the specified namespace.

```
curl -ivk -X GET -H "Content-Type: application/x-www-form-urlencoded" -H "Authorization: Bearer $[***MY-TOKEN-NAME***]" https://[***DATA LAKE-
```

```
LOADBALANCER***]/[***DATA LAKE-NAME***]/cdp-share-access/hms-api/icecli/v1/
namespaces/<namespace>/tables
```

Example:

```
My-token-name=$(curl -k -X POST -H "Content-Type: application/x-www-form-ur
lencoded" -d "client_id=b9efc3dd-3695-4867-9e4c-523389c8a78b&client_secret=W
WpsbFptTXpaRlF0TXpZNU5TMDBPRFkzTFRsbE5HTXROVEl6TXpnNVl6aGhOemhpOjpNRGt5Tnpne
ElHUXRZak00TmkwMFpqTXhMVGczTTJZdFptTXhOekl6TnpVNFlqazI=&grant_type=client_cr
edentials" "https://apr24-LB-my-datalake-load-balancer.elb.us-west-2.amazona
ws.com/my-datalake-name/cdp-share-access/hms-api/icecli/v1/oauth/tokens" |
jq -r '.access_token')
```

```
curl -ivk -X GET -H "Content-Type: application/x-www-form-urlencoded" -H "Au
thorization: Bearer $My-token-name" https://apr24-LB-my-datalake-load-balanc
er.elb.us-west-2.amazonaws.com/my-datalake-name/cdp-share-access/hms-api/ice
cli/v1/namespaces/hive_rest_airline_orc/tables
```

Load Tables: /v1/{prefix}/namespaces/{namespace}/tables/{table}

Load a table from the catalog.

```
curl -ivk -X GET -H "Content-Type: application/x-www-form-urlencoded" -
H "Authorization: Bearer $[***MY-TOKEN-NAME***]" https://[***DATA LAKE-
LOADBALANCER***]/[***DATA LAKE-NAME***]/cdp-share-access/hms-api/icecli/v1/
namespaces/<namespace>/tables/icebergtable
```

Example:

```
My-token-name=$(curl -k -X POST -H "Content-Type: application/x-www-form-ur
lencoded" -d "client_id=b9efc3dd-3695-4867-9e4c-523389c8a78b&client_secret=W
WpsbFptTXpaRlF0TXpZNU5TMDBPRFkzTFRsbE5HTXROVEl6TXpnNVl6aGhOemhpOjpNRGt5Tnpne
ElHUXRZak00TmkwMFpqTXhMVGczTTJZdFptTXhOekl6TnpVNFlqazI=&grant_type=client_cr
edentials" "https://apr24-LB-my-datalake-load-balancer.elb.us-west-2.amazona
ws.com/my-datalake-name/cdp-share-access/hms-api/icecli/v1/oauth/tokens" |
jq -r '.access_token')
```

```
curl -ivk -X GET -H "Content-Type: application/x-www-form-urlencoded" -H "Au
thorization: Bearer $My-token-name" https://apr24-my-datalake-load-balancer.
elb.us-west-2.amazonaws.com/my-datalake-name/cdp-share-access/hms-api/icecli
/v1/namespaces/hive_rest_airline_orc/tables/airport_iceberg_external
```

Related Information

[Sample Spark workload to access data](#)

[REST Catalog API calls throughput](#)

[Iceberg REST Catalog API specification](#)