

Cloudera Runtime 7.3.1

Migrating Sentry to Ranger for Cloudera on premises

Date published: 2020-07-28

Date modified: 2024-12-10

The Cloudera logo is displayed in a bold, orange, sans-serif font. The word "CLOUDERA" is written in all caps, with the letter 'E' in "CLouDERA" featuring a horizontal line through its center.

<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

Migrating from Sentry to Ranger.....	4
Consolidating policies created by Authzmigrator.....	5
Customizing the authorization-migration-site.xml file.....	6
Check MySQL isolation configuration.....	8
Ranger policies allowing create privilege for Hadoop_SQL databases.....	8
Ranger policies allowing create privilege for Hadoop_SQL tables.....	9
Access required to Read/Write on Hadoop_SQL tables using SQL.....	11
Mapping Sentry permissions for Solr to Ranger policies.....	11
Configuring Ranger ACL Sync.....	12
Migrate permissions manually using the authzmigrator tool.....	13
Authzmigrator tool.....	14
Exporting Permissions from Sentry Server.....	14
Ingesting permissions into Ranger.....	15

Migrating from Sentry to Ranger

Before deciding to migrate from Sentry to Ranger, read the Sentry to Ranger Concise Guide and the topics in this guide.

The [Sentry to Ranger Concise Guide blog post](#) describes fundamental differences between Sentry and Ranger, compares the two products, and provides additional information that helps prepare you for your migration.

Read the topics in this section for information about preparing to migrate Sentry permissions to Ranger policies and topics that describe how to migrate once you are ready.

Sentry (CDH) had an object ownership feature, which added ownership permissions for all the databases/tables created. This feature was added in CDH-5.16 and supported through CDH-6.2. After enabling the ownership feature Sentry would grant owner permission for all the databases/tables created after enablment.

Ranger default policies for Hadoop Sql

Policy Name	User	Permissions
all - database, table, column	{OWNER}	all permissions
all - database, table	{OWNER}	all permissions
all - database, udf	{OWNER}	all permissions
all - database	{OWNER}	all permissions

After migration from Sentry:

- All the users who have {OWNER} permissions on objects, such as databases/tables, will get All the permissions from above default Ranger policies.
- Above Ranger policies will be applicable only to objects for whom they are the owner.
- Even if Sentry does not have owner mapping, in other words, the ownership feature is disabled, this scenario holds true.
- Sentry roles that are empty and have no privileges are not copied over to Ranger after upgrade, like roles created with Sentry enabled.

For example:

```
CREATE ROLE <role name>;
but no grants for the role executed like below:
GRANT <privilege> on <object> TO ROLE <ROLE>;
Wont be migrated.
```



Note: If you are using MySQL as the Ranger database and seeing the following error while migrating from Sentry to Ranger:

Error:

```
com.mysql.cj.jdbc.exceptions.MySQLTransactionRollbackException: Lock wait timeout exceeded; try restarting transaction
```

Resolution

1. In Cloudera Manager Configuration Search , type core-site.xml, then click Search.
2. In CORE_SETTINGS-1 (Service-Wide), in Name, type ranger.client.pool.size.
3. In Value, type 1.
4. Click Save Changes (CTRL+S).
5. On the Cluster Actions , click Restart.

Consolidating policies created by Authzmigrator

Before migrating/importing Sentry permission into Ranger policies, add the `authorization.migration.skip.owner.policy = true` configuration in `authorization-migration-site.xml`.

Migrating Sentry - Ranger with Ownership Feature Enabled

Scenario:

Ownership feature enabled in Sentry. After enabling ownership feature, Sentry would have owner permission for all the databases/tables created after enabling this feature.

Case:

Bob created 1000 tables `tab1`, `tab2`, `tab2 ... tab1000` under database `demoDB`.

Since ownership feature enabled, Sentry will have OWNER-MAPPING.

After migration from Sentry to Ranger using Authzmigrator tool Ranger will have 1000 OWNER policies for each mapping in sentry. However, Ranger already has default OWNER policies. There is no need to have individual table level OWNER policies. These 1000 policies will be hard to administer from the Ranger UI.

Skipping OWNER policy creation for each every OWNER-MAPPING in Sentry.

Add `authorization.migration.skip.owner.policy = true` to `authorization-migration-site.xml` to avoid creating so many policies in Ranger during migration.



Note: Beginning in 7.1.7 sp2, you can enable / alter the value for the `authorization.migration.skip.owner.policy` property from Cloudera Manager, during install.

```
<property>
  <name>authorization.migration.skip.owner.policy</name>
  <value>true</value>
</property>
```

Customizing the authorization-migration-site.xml file

You can customize the default behavior of the Sentry to Ranger policy migration, using a safety valve in Cloudera Manager.

About this task

Ranger configurations expose a safety-valve for authorization-migration-site.xml to allow users to customize properties that control migration of policies from Sentry to Ranger. Ranger embeds a default set of configurations in authorization-migration-site.xml, for example,

```
authorization.migration.export.output_file = hdfs:///user/sentry/export-permissions/permissions.json
authorization.migration.ingest.is_dry_run = false
authorization.migration.role.permissions = true
authorization.migration.translate.url.privileges = false
authorization.migration.ingest.merge.ifexists = true
authorization.migration.export.target_services = HIVE,KAFKA
authorization.migration.migrate.url.privileges = true
authorization.migration.export.migration_objects = ""
authorization.migration.object.filter = ""
```

You can customize these configurations, using the Ranger Admin Advanced Configuration Snippet (Safety Valve) for conf/authorization-migration-site.xml "safety valve" in Cloudera Manager.

For example, setting the values of the following properties is required to update the location prefix in all URI privileges during the import:

```
authorization.migration.translate.url.privileges = true
authorization.migration.destination.location.prefix = hdfs://<new_cdp_name>service
```



Important:

Importing large policy sets requires extra time for the utility script to complete. For large policy sets, you can increase the default (7200 s) timeout setting for the import script, by adding the following configuration property and value to Ranger Admin Advanced Configuration Snippet (Safety Valve) for conf/authorization-migration-site.xml "safety valve":

```
authorization.migration.ingest.timeout.sec = 18000
```

To customize properties:

Procedure

1. In Cloudera Manager Configuration Search type authorization-migration-site.xml, then click Search.
2. In Ranger-1 > Ranger Admin Default Group, click +(Add).
3. In Name, type a property name, such as authorization.migration.translate.url.privileges.
4. In Value, type a property value, such as true.
5. Click Save Changes.
6. Repeat steps 2-5 for each property that you want to customize.

Results

Each property/value pair that you save adds a property or overwrites the default value assigned to that property in the authorization-migration-site.xml file.

Currently, while running the Importing Sentry privileges into Ranger policies step to import the old Sentry grants to Ranger, with the following configurations in the Ranger Admin Advanced Configuration Snippet (Safety Valve) for `conf/authorization-migration-site.xml`:

```
authorization.migration.translate.url.privileges=true
```

and

```
authorization.migration.destination.location.prefix=[hdfs://ns1]
```

The `file://` Sentry URI grants are created as `hdfs://` URL policies in Ranger.

For example:

```
file:///opt/cgfiles/common/jdbc/my_udf-0.2.2.jar
```

becomes

```
[hdfs://ns1/opt/cgfiles/common/jdbc/my_udf-0.2.2.jar]
```

By using the `authorization.migration.url.ignore.scheme` configuration you can add multiple, comma-separated file system prefixes. The values provided in config will not update to prefix provided in property `authorization.migration.destination.location.prefix` while importing Sentry privileges into Ranger policies.

In case, if `authorization.migration.translate.url.privileges=true`

and

`authorization.migration.destination.location.prefix=[hdfs://ns1]` are already set and if we set

`authorization.migration.url.ignore.scheme = file`, then any url policy with file prefix would not be replaced by `hdfs://ns1` during import.

For example:

```
file:///opt/cgfiles/common/jdbc/my_udf-0.2.2.jar
```

remains

```
file:///opt/cgfiles/common/jdbc/my_udf-0.2.2.jar
```

Currently during `AuthzMigrator Export`, all Sentry data (Dbs/Tbls/Urls) are exported from sentry to `permission.json`.

There is an option to export Sentry data only for given Hive objects (databases and tables and the respective URLs).

You can use the `authorization.migration.export.migration_objects` configuration property in `authorization-migration-site.xml` to provide Hive object details at the time of Sentry export.

While providing configuration value, use the following format:

- single database `#db={db_name}` eg. `db=dio_work`
- single table `#db=dio_work/tbl=ur_cdp_upgrade_ext` (database and table should be separated by /)
- multiple databases `#db=dio_work/tbl=.*,db=dio_work_2/tbl=.*` (databases should be comma separated)
- multiple tables `#db=dio_work/tbl=ur_cdp_upgrade_ext,db=dio_work/tbl=ur_cdp_upgrade_mngd`
- all tables of database `#db=dio_work/tbl=.*`
- all databases and all tables `#db=.*tbl=.*`

For example:

```
authorization.migration.export.migration_objects = db=dio_work/tbl=ur_cdp_upgrade_ext,db=dio_work/
tbl=ur_cdp_upgrade_mngd
```

**Note:**

From the Cloudera Manager UI you can set configurations in authorization-migration-site.xml by clicking Replication Replication Policies Create Replication Policy Hive External Replication Policy .

If Permissions is selected as either If Sentry permissions were exported from the CDH cluster, import both Hive object and URL permissions. or If Sentry permissions were exported from the CDH cluster, import only Hive object permissions., then the Sentry-Ranger migration tab will be visible.

In the Sentry-Ranger Migration tab, you can set configuration parameters.

Check MySQL isolation configuration

Before migrating a MySQL database for CDH Sentry to CDP Ranger, you must check and set isolation configuration to READ-COMMITTED.

Before you begin

CDP Ranger MySQL database must have isolation set to READ-COMMITTED.

About this task

You must check the isolation configuration of the MySQL database used for CDH Sentry before migrating to Ranger. IF the isolation configuration for CDH Sentry setting is REPEATABLE-READ, you must change the isolation setting to READ-COMMITTED.

Procedure

1. Log in to MySQL server.
2. Run the following query:

```
SELECT @@GLOBAL.tx_isolation, @@tx_isolation, @@session.tx_isolation;
```

- a) If the query output is:

```
-----
--
@@GLOBAL.tx_isolation | @@tx_isolation | @@session.tx_isolation
-----
-----
REPEATABLE-READ | REPEATABLE-READ | REPEATABLE-READ
```

- b) Then, set the isolation to READ-COMMITTED,using the following query:

```
mysql> SET tx_isolation = 'READ-COMMITTED';
Query OK, 0 rows affected (0.00 sec)

mysql> SET GLOBAL tx_isolation = 'READ-COMMITTED';
Query OK, 0 rows affected (0.00 sec)
```

Ranger policies allowing create privilege for Hadoop_SQL databases

Users with authorized access through Ranger policies in Hadoop SQL with at least one of the following permissions can create databases.

In CDP, an authorized user can create Hadoop_SQL databases with or without specifying location. If you do not specify location, then the database is created in the default HDFS location used by Hadoop_SQL. If you specify location, then the database is created in the HDFS location you specify.

- A user creating databases with location clauses requires one of the following additional access:
 - direct read and write access to the HDFS location
 - a Ranger Hadoop_SQL URL policy that provides the user all permissions on the HDFS location
- A hive user creating databases with location clauses must have all permissions on the HDFS location using one of the following:
 - an appropriate HDFS POSIX permission
 - HDFS ACL
 - HDFS Ranger policy



Note: If you choose to use an HDFS Ranger policy for this purpose, make sure to refer to the HDFS location in the Ranger policy using a path, such as: /databases/sample/username, not a URL, such as: hdfs://nameservice1/databases/sample/username .

Table 1: Permissions allowing a user to create a database

User	Permission	Database	Table	Column	UDF
hive and impala	all	all (database=*)			
		all (database=*)	all (table=*)		
		all (database=*)	all (table=*)	all (column=*)	
		all (database=*)			udf=*
hive and impala	create	all (database=*)			
		all (database=*)	all (table=*)		
		all (database=*)	all (table=*)	all (column=*)	
		all (database=*)			udf=*



Note:

- For use-cases where only create access is provided and drop access is not provided explicitly, the user might implicitly get a few other permissions through the default policies added (unless the default policies are modified).
- The default all database and all database, table policy usually would list {OWNER} as an authorized user.
- Removing {OWNER} from these default policies would restrict access to users with specific permissions listed explicitly in policies. Removing {OWNER} is not recommended. Proceed with caution when considering such an action.

Related Information

[Resource-based Services and Policies](#)

Ranger policies allowing create privilege for Hadoop_SQL tables

Users with authorized access through Ranger policies in Hadoop SQL with at least one of the following permissions can create external or managed tables on the corresponding database(s) listed in the policy.

- A user creating external tables with location clauses requires one of the following additional access:
 - direct read and write access to the HDFS location
 - a Ranger Hadoop_SQL URL policy that provides the user read and write permissions on the HDFS location
- A user creating external tables with location clauses must have read and write permissions on the HDFS location using one of the following:
 - an appropriate HDFS POSIX permission
 - HDFS ACL
 - HDFS Ranger policy



Note: If you choose to use an HDFS Ranger policy for this purpose, make sure to refer to the HDFS location in the Ranger policy using a path, such as: /databases/sample/username, not a URL, such as: hdfs://nameservice1/databases/sample/username . Make sure that the URL defined in Ranger does not have a trailing /.

Table 2: Permissions allowing a user to create a table

User	Permission	Database	Table	Column	UDF
hive and impala	all	database=* or <database name>			
		database=* or <database name>	all (table=*)		
		database=* or <database name>	all (table=*)	all (column=*)	
		database=* or <database name>			udf=*
hive and impala	create	database=* or <database name>			
		database=* or <database name>	all (table=*)		
		database=* or <database name>	all (table=*)	all (column=*)	
		database=* or <database name>			udf=*



Note:

- For use-cases where only create access is provided and drop access is not provided explicitly, the user might implicitly get a few other permissions through the default policies added (unless the default policies are modified).
- The default all database and all database, table policy usually would list {OWNER} as an authorized user.
- For these use-cases where only permissions were provided at the database and udf levels, the user may still be able to create tables due to the reasons specified above.
- Removing {OWNER} from these default policies would restrict access to users with specific permissions listed explicitly in policies. Removing {OWNER} is not recommended. Proceed with caution when considering such an action.
- Any managed table creation using an external location would fail with the following error: A managed table's location should be located within managed warehouse root directory or within its database's managedLocationUri.

Related Information

[Resource-based Services and Policies](#)

Access required to Read/Write on Hadoop_SQL tables using SQL

Users with authorized access through Ranger policies in Hadoop SQL with at least one of the following permissions can read and write to external or managed Hadoop_SQL tables using SQL syntax.

- Any user who created a managed or an external table {owner} can select and insert data in the table, provided the permissions for database, table and columns are present in Hadoop_SQL service within Ranger.
- Any user with select privileges on columns, tables and databases in Hadoop_SQL service within Ranger can read data from managed or external tables by executing SQL statements.
- Any user with update privileges on columns, tables and databases in Hadoop_SQL service within Ranger can write data into Managed or External Tables by executing SQL statements.

Related Information

[Resource-based Services and Policies](#)

Mapping Sentry permissions for Solr to Ranger policies

Use the mapping reference table to create Ranger policies that reflect the privileges defined for Solr in your Sentry permissions.

Sentry has the following objects for Solr:

- admin
- collection
- config
- schema

The admin object type controls access to administrative actions through the following privilege objects:

- collection
- cores
- security
- metrics
- autoscaling

Ranger has only one object right now, which is collection. Permissions for collections are of type:

- SolrAdmin
- Query
- Update
- Other

Table 3: Ranger policies required to set equivalent access that Sentry privileges allowed

Sentry privilege	Ranger policy
	Collections
admin=collections - action=UPDATE collection=<aliasName> - action=UPDATE	All collections - permission SolrAdmin

Sentry privilege	Ranger policy
admin=collections - action=UPDATE collection=<collectionName> - action=UPDATE	Policy for <collectionName>, permissions: SolrAdmin
admin=collections - action=UPDATE	All collections - permissions: SolrAdmin
admin=collections - action=QUERY collection=<collectionName> - action=QUERY	Policy for <collectionName> - permissions: SolrAdmin
Cores	
admin=cores - action=UPDATE collection=<coreName> - action=UPDATE	All collections - permission: SolrAdmin
admin=cores - action=QUERY collection=<coreName> - action=QUERY	All collections - permission: SolrAdmin
Configs	
config=<configName> - action=*	All collections = permission: SolrAdmin
Non-Administrative	
collection=<collectionName> - action=QUERY	Policy for <collectionName> - permissions: Query, Others
collection=<collectionName> - action=UPDATE	Policy for <collectionName> - permissions: Update

Configuring Ranger ACL Sync

You can configure Ranger ACL sync to view user access permissions in a manner similar to the HDFS `getfacl` command after migrating from CDH to CDP.

Introduction

Ranger ACL Sync feature replaces a similar feature previously supported in Sentry. Ranger ACL-Sync is implemented through RMS module which consists of RMS server and chained-plugin and corresponding storage plugin - HDFS/Ozone. Ranger ACL Sync supports viewing access permissions on storage locations across:

- Ranger policies for storage locations
- Ranger policies for Hive entities
- any native storage ACLs

Ranger ACL Sync is considered essential for those customers migrating from CDH to CDP clusters who also used Sentry ACL-sync feature.

Ranger ACL Sync addresses the following customer requirements to support:

- an operational need for end-users with no access to Ranger-admin to check the permissions on a HDFS resource
- a tool-set that can verify if the Sentry access permissions are correctly migrated

Sentry implemented an HDFS command `getfacl` that provided access permissions on a given resource to the end-user in a structured, easy-to-understand form. Ranger ACL-sync is designed to offer the same user experience to the end-users.

Design

The Ranger policy-engine supports an API that, for a resource, returns a static view of the permissions allowed for all users, groups and roles. The view of the access permissions is specific to the component's access model. In other words, for Hive resource the permissions as defined in the Hive service definition, and for HDFS resource, the view consists of read/write/execute permissions. This functionality is leveraged further, by extending it to chained plugins.

Ranger ACL-sync internally chains HDFS plugin with Hive plugin to provide composite evaluation of access for an HDFS location that maps to a Hive entity. With this extension, the static determination of accesses for a resource considers any chained plugins (and corresponding policy-engine) in its computation. This also requires conversion between Hive access types and HDFS access types, and combining of accesses by multiple (in this case HDFS and Hive) plugins.

The front-end for accessing this static permission map is the `getfacl` command supported by HDFS. Using this front-end obviates the need for any additional configuration set-up (other than one that enables/disables this feature). It also gives Sentry users (almost) identical experience when interrogating accesses supported for a resource.

Configuration

There are two new configuration parameters introduced for this feature. Both are specified using the safety valve mechanism on HDFS (`ranger-hdfs-security.xml`).

1. Go to Cloudera Manager HDFS Configuration .
2. In Search, type `ranger-hdfs-security.xml`.
3. In HDFS Service Advanced Configuration Snippet (Safety Valve) for `ranger-hdfs-security.xml`, add the following properties/values:

`is.facl.interception.enabled=true`

(Default:false): If true, this feature is activated.

`ranger.plugin.hdfs.policyengine.option.disable.role.resolution=true`

(Default:true) : If false, the roles are resolved to their constituent users and groups for reporting the access permissions.

4. Click Save Changes (CTRL+S).
5. Restart services with stale configurations.

Limitations

Ranger evaluates access based on static and dynamic (time-based, accessing IP-based, condition-based) policies. The static view of the access permissions given only a resource, is not a complete and always accurate picture. However, for the customers moving from Sentry to Ranger, this may not be an issue.

Related Information

[Ranger RMS - HIVE-HDFS ACL Sync Overview](#)

[Configure Hive-HDFS / Hive-OZONE ACL Sync](#)

Migrate permissions manually using the authzmigrator tool

To migrate Hive object and URL permissions and Kafka permissions from CDH to Cloudera Private Cloud Base, you can use the `authzmigrator` tool. The tool exports the Hive/Impala and Kafka RBAC Sentry permissions in the CDH cluster to a JSON file, and then converts and ingests it into Ranger in the Cloudera Private Cloud Base 7 cluster. You must migrate Solr permissions manually from the CDH cluster to the Cloudera Private Cloud Base 7 cluster.

You can use one of the following methods to upgrade a CDH cluster to a Cloudera cluster and to migrate the Sentry permissions in CDH to Ranger in Cloudera:

- Upgrade wizard - You export the permissions in the CDH cluster and after the in-place upgrade completes, you import the permissions into Ranger.
- Side-by-side migration (side-car migration) - You bring up a second environment that is a Cloudera environment. You then move all the data and metadata to this new environment. You can either use the `authzmigrator` tool to migrate Hive object and URL permissions and Kafka permissions to Ranger or run a Hive replication policy in the Replication Manager to migrate the Hive object and URL permissions to Ranger.

Authzmigrator tool

To migrate Hive object and URL permissions and Kafka permissions using the authzmigrator tool, you must export the permissions in the Sentry server to a file on the source cluster (for example, CDH cluster) and then ingest the file into Ranger service on the target cluster (for example, Cloudera cluster).

Exporting Permissions from Sentry Server

Use the authzmigrator tool to export the Sentry permissions from the Sentry server on the source cluster to a file.

About this task

During side-by-side migration (side-car migration), you can use the authzmigrator tool to migrate the Hive object and URL permissions and Kafka permissions to Ranger.

Procedure

1. Download the `authz_export.tar.gz` file and extract it. For information about downloading the file, contact *Cloudera Support*.

The `authz_export.tar.gz` file contains directories named `jars` and `config`. It also has an `authz_export.sh` file. The `config` directory contains default configurations that you can use for reference.

2. Replace the `sentry-site.xml` and `core-site.xml` in the `config` directory with the configuration files from the Sentry directory on the Sentry server in the source cluster.

For example, the Sentry directory on the source cluster is located in the `/var/run/cloudera-scm-agent/process/<sentry-service>/` location.

3. Edit the `sentry-site.xml` file to perform the following steps:

- a) Update the database username and password for the Sentry database with the following credentials:

```
sentry.store.jdbc.user
```

```
sentry.store.jdbc.password
```

- b) Remove the `hadoop.security.credential.provider.path` property in the file.

4. Edit the `core-site.xml` file to perform the following steps:

- a) Update the value for the property `fs.defaultFS` to `file:///`.

- b) Remove the `hadoop.security.credential.provider.path` property in the file.

5. In the `authorization-migration-site.xml` file in the `config` directory, perform the following steps:

- a) Make sure that the `authorization.migration.export.target_services` property has the list of services for which the permissions are to be exported.

Valid values include: HIVE KAFKA KUDU

- b) Update the information in the `authorization.migration.export.output_file` property to the absolute location of the file where permissions should be exported.

6. Verify whether the Java execution path for the Sentry server and the `JAVA_HOME` property in the `authz_export.sh` script matches. To verify the path and property, perform the following tasks:

- a) To locate the Java execution path that Sentry server uses, run the `ps aux | grep org.apache.sentry.SentryMain` command.

- b) If the path is not `/user/java/default/bin/java`, edit the `authz_export.sh` script, add the path that the Sentry server uses to the `JAVA_HOME` property, and save the file.

For example, if the Sentry server uses the `/usr/java/jdk1.8.0_141-cloudera/bin/java` path, change the `JAVA_HOME` property in the `authz_export.sh` script to `/usr/java/jdk1.8.0_141-cloudera`.

7. Run the `authz_export.sh` script using the `sh authz_export.sh` command.

```
$ sh authz_export.sh
log4j:WARN No such property [conversionPattern] in org.apache.solr.util
.SolrLogLayout.
389 T1 cceacv.CommonConfigurationValidator.validateServiceAndAuthzObjects
No migration objects specified. Migrate all privileges
451 T1 cceacv.CommonConfigurationValidator.validatePathScheme No matching
schemes with [[file, hdfs, s3a, s3, abfs]]. Use default scheme [file]
806 T1 odu.Log4JLogger.info Property datanucleus.cache.level2 unknown -
will be ignored
3066 T1 odu.Log4JLogger.warn WARN Metadata has jdbc-type of null yet this
is not valid. Ignored
4266 T1 ccea.PolicyStore.lambda$fetchPermissionInfo$1 Fetching permissi
ons for all SQL objects for service type [HIVE]
4375 T1 ccea.PolicyStore.fetchDbPermissionInfo Total privileges retrieved
[8]
4401 T1 ccea.PolicyStore.lambda$getRoleGroupMap$4 Fetched 3 roles
4403 T1 ccea.PolicyStore.lambda$getRoleGroupMap$4 Roles to group mapping
retrieved [3]
4479 T1 ccea.PolicyStore.fetchKafkaPermissionInfo No privileges retrieved
for export
4484 T1 ccea.SentryExportTask.constructPolicyList [Completed] Constructing
policy list
4561 T1 ccea.FSClient.write [Started] Writing policy information to /opt/
backup/permissions.json
4578 T1 oahu.NativeCodeLoader.<clinit> WARN Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
4596 T1 ccea.FSClient.write [Completed] Writing policy information to /opt
/backup/permissions.json
4597 T1 ccea.SentryExportTask.execute Exporting permission information to
location /opt/backup/permissions.json successful
4597 T1 ccea.Main.main Exporting the permissions is complete
```

The permissions are exported to the `/opt/backup/permissions.json` file.

What to do next

You can ingest the permissions into Ranger.

Ingesting permissions into Ranger

After you run the `authzmigrator` tool to export the permissions on the source cluster, ingest them into the Ranger service on the target cluster.

About this task

To ingest the permissions in the `/opt/backup/permissions.json` file, perform the following steps:

Procedure

1. In the target cluster, copy the `/opt/backup/permissions.json` file to the `/user/sentry/export-permissions/permissions.json` HDFS location.
2. Make sure the user `ranger` has read-write permissions on the file. To set the read-write permissions for the `ranger` user, run the following command:

```
hdfs dfs -setfacl -m -R user:ranger:rwx /user/sentry/export-permissions/permissions.json
```
3. Log in to Cloudera Manager.

4. In the Ranger service, click `Actions Import Sentry permissions` .

The Ranger service ingests the permissions present in the `/user/sentry/export-permissions/permissions.json` file. You can view the permissions in the Ranger Admin UI.