Cloudera Runtime 7.3.2

Authorization reference

Date published: 2020-07-28
Date modified: 2026-03-31

https://docs.cloudera.com/

Legal Notice

Contents

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 permssions 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.

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.MySQLTransactionRollbac
kException: 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-perm
issions/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_namese
rvice>
```

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.timout.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 Cloudera Ranger, you must check and set isolation configuration to READ-COMMITTED.

Before you begin

Cloudera 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 audit schema reference

See the following link to the current version of the Ranger Audit Schema:

Ranger Audit Schema: https://cwiki.apache.org/confluence/display/RANGER/Ranger+Audit+Schema#RangerAuditSchema-AudittoHDFS

Ranger database schema reference

See the following links to versions of the Ranger Database Schema:

Ranger v2.0.0 : https://cwiki.apache.org/confluence/display/RANGER/Ranger+2.0.0+Database+Schema

Ranger v2.1.0 : https://cwiki.apache.org/confluence/display/RANGER/Ranger+2.1.0+Database+Schema

Ranger v2.2.0 : https://cwiki.apache.org/confluence/display/RANGER/Ranger+2.2.0+Database+Schema

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 Cloudera, 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 extermal 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 extermal 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
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	

Sentry privilege	Ranger policy
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