116

DBMS_KAFKA_ADM

The `DBMS_KAFKA_ADM` package provides a PL/SQL interface to create cluster definitions, which you can then use to grant access to Kafka cluster data for applications.

Administrators granted the `OSAK_ADMIN_ROLE` can use `DBMS_KAFKA_ADM` package to create applications that query Kafka data from Oracle Database views and tables.

DBMS_KAFKA_ADM Overview

The `DBMS_KAFKA_ADM` package enables you to manage the Oracle SQL access to a Kafka cluster.

To set up access to Kafka topics, you require the privileges granted with `OSAK_ADMIN_ROLE`. You use these privileges to provide a cluster definition that helps you to manage resources when connecting to Kafka clusters. The ability of users to share a cluster definition enables the underlying framework to share connections to the clusters. Sharing connections conserves memory and network resources.

DBMS_KAFKA_ADM Security Model

Oracle recommends that you grant the `OSAK_ADMIN_ROLE` in the target Oracle Database to an administrator user for Oracle SQL Access to Kafka.

Using the `OSAK_ADMIN_ROLE` simplifies the process of granting system privileges required to create and administer Oracle SQL access to Kafka applications. Also, administrators with the `OSAK_ADMIN_ROLE` grant access for other users to the Oracle Database object that has the relevant Kafka view and table for which they need access.

In addition to the `OSAK_ADMIN_ROLE`, an administrator for Oracle SQL access to Kafka applications must have the following system privileges:

- `CREATE SESSION`
- `ALTER SESSION`

- `CREATE CREDENTIAL`, to create a Kafka SASL-SSL (Simple Authentication and Security Layer) password or OSS (Oracle Streaming Service) authToken
- `CREATE ANY DIRECTORY`, to create cluster access and cluster configuration directory
- `DROP ANY DIRECTORY`, to drop cluster access and cluster configuration directory
- `READ ON` privileges to `sys.dbms_kafka_clusters`
- `READ ON` privileges to `sys.dbms_kafka_applications`
- `READ ON` privileges to `sys.dbms_kafka_messages`
- `EXECUTE ON` privileges to `sys.dbms_kafka_adm`

DBMS_KAFKA_ADM Constants

The `DBMS_KAFKA_ADM` package is a stateful package that uses the constants described here as part of the declared package state in the package specification.
The `DBMS_KAFKA_ADM` constants include constants with literal initial values and constants that you specify in your package build.

Kafka Providers

The provider of the Kafka server can be either Apache, or Oracle Cloud Infrastructure (OCI) Oracle Streaming Service (OSS).

```
KAFKA_PROVIDER_APACHE CONSTANT VARCHAR2(6) := 'APACHE';
KAFKA_PROVIDER_OSS CONSTANT VARCHAR2(3) := 'OSS';
```

States of a cluster definition

```
STATE_CONNECTED CONSTANT INTEGER := 0;
STATE_MAINTENANCE CONSTANT INTEGER := 1;
STATE_BROKEN CONSTANT INTEGER := 2;
STATE_DEREGISTERED CONSTANT INTEGER := 3;
```

Kafka connection modes

Kafka connection modes specify one of the following connection mode constants:

- High Throughput

  If the connection mode (`connmode`) is `high_throughput`, then the applications associated with this connection require all of the data to be delivered as fast as possible. This connection mode is optimal for use from STREAMING and LOADING applications.

  For example:

  ```
  CONNECTION_MODE_HI_THRU CONSTANT VARCHAR2(15) := 'high_throughput';
  ```

- Low Latency

  If the connection mode (`connmode`) is `low_latency`, then the underlying layer attempts to return the first rows as fast as possible. This connection mode is most useful for applications that use only a small amount of Kafka data for each load.

For example:

```
CONNECTION_MODE_LO_LAT CONSTANT VARCHAR2(11) := 'low_latency';
```

- Options

```
OPT_CONNECTION_MODE CONSTANT VARCHAR2(30) := 'connmode';
```

Summary of DBMS_KAFKA_ADM Procedures

`DBMS_KAFKA_ADM` procedures enable you to create, delete, and configure Oracle SQL access to Kafka.

Kafka providers

Kafka provider constants are used to define the Kafka cluster. Because the security models are different between an Apache Kafka-based cluster and an Oracle Streaming Service-based cluster, you must define to which type of Kafka environment you are connecting, so that the correct secure connection parameters can be passed.
For example:

```
KAFKA_PROVIDER_APACHE CONSTANT VARCHAR2(6) := 'APACHE';
KAFKA_PROVIDER_OSS CONSTANT VARCHAR2(3) := 'OSS';
```

States of a cluster definition

```
STATE_CONNECTED CONSTANT INTEGER := 0;
STATE_MAINTENANCE CONSTANT INTEGER := 1;
STATE_BROKEN CONSTANT INTEGER := 2;
STATE_DEREGISTERED CONSTANT INTEGER := 3;
```

Kafka connection modes

Kafka connection modes specify the Kafka rules that Oracle SQL access to Kafka follows to read a batch of records. Depending on the requirements for your application, you can specify either a low latency rule or a high throughput rule, depending on which option is the more efficient use of resources. You should consider tradeoffs between throughput and latency when you create your connections. The latency between the time that a record is generated and the time that you want that data made available to your application should determine the whether you make low latency your priority, or high throughput your priority.

If your application requires data access in near real-time, so that your application can react to data within seconds or a minute of its being generated by the Kafka cluster, then you should choose low latency connections. With a low latency connection, the Oracle SQL Kafka connection buffers are set to a smaller size, and the reads to the Kafka topic are made frequently, so that the delay between the time a record is generated to the Kafka topic and consumed by the application using Oracle SQL access to Kafka is within the performance requirements of your application. Each read from the Kafka topic is expected to consist of a relatively few number of records, and the number of reads from the Kafka cluster are high. This form of Kafka connection is particularly suited to updating an application with near real-time concurrent transactions, or where the application must monitor and react to changes quickly. However, this configuration can be resource-intensive, as the number of transactions are relatively high for the amount of data received in each transaction.

If your application does not have near real-time requirements, so that your application can delay updates in favor of processing a large batch of records, in a cadence of 15 minutes, or an hour, or every 8 hours, then a high throughput connection can be a more efficient choice. With a connection configured for high throughput, each batch of Kafka data is streamed more infrequently, but the amount of data streamed in each transaction is larger. Accordingly, the Oracle SQL access to Kafka buffer size is larger, and the network I/O and memory allocations are configured for these larger batches. Because these transactions are streamed less frequently, the overall amount of server resources consumed by the Oracle SQL access to Kafka transaction is less

- Low Latency

```
CONNECTION_MODE_LO_LAT CONSTANT VARCHAR2(11) := 'low_latency';
```

- High Throughput

```
CONNECTION_MODE_HI_THRU CONSTANT VARCHAR2(15) := 'high_throughput';
```

- Options

```
OPT_CONNECTION_MODE CONSTANT VARCHAR2(30) := 'connmode';
```

This table lists the DBMS_KAFKA_ADM procedures, and briefly describes them.

Table 116-1 DBMS_KAFKA_ADM Package Procedures

Subprogram	Description
CHECK_CLUSTER	Tests the Kafka cluster connectivity.
DEREGISTER_CLUSTER	Deregisters the Kafka cluster.
DISABLE_CLUSTER	Disables the Kafka cluster.
ENABLE_CLUSTER	Enables the Kafka cluster.
REGISTER_CLUSTER	Registers the Kafka cluster.
UPDATE_CLUSTER_INFO	Updates the host server, or Kafka cluster options, or both.

CHECK_CLUSTER

The `DBMS_KAFKA_ADM` procedure `CHECK_CLUSTER` tests the Kafka cluster connectivity.
Use this procedure to confirm that a connection can be established with the configured security information. The function returns the state of the cluster.

Syntax

```
FUNCTION CHECK_CLUSTER(
                   cluster_name    IN VARCHAR2
  ) RETURN INTEGER;
```

Parameters

Table 116-2 CHECK_CLUSTER procedure parameters for DBMS_KAFKA_ADM

Parameter	Description
cluster_name	Name of an existing Kafka cluster Case-insensitive.

Usage Notes

As the Oracle SQL access to Kafka cluster administrator, you should check the connectivity of the cluster. Checking the connectivity ensures that the bootstrap server list and the security related configuration that you have provided for the Kafka cluster is proper, and that a connection can be successfully established. You can use this function to test the connection when you run `REGISTER_CLUSTER`, and the security configuration has been established. After the cluster connection has been verified, you can make the cluster definition available to be used by user applications.

You also use this function if users have reported issues when calling `DBMS_KAFKA.LOAD_TEMP_TABLE` and `DBMS_KAFKA.EXECUTE_LOAD_APP`, both of which select from Oracle SQL access to Kafka views.

Examples

Suppose you have completed your initial cluster definition for the cluster ExampleCluster, and registered the cluster. Next, you use this procedure to check the configuration:

```
EXEC SYS.DBMS_OUTPUT.PUT_LINE (
           SYS.DBMS_KAFKA_ADM.CHECK_CLUSTER ('ExampleCluster'));
```

DEREGISTER_CLUSTER

The `DBMS_KAFKA_ADM` procedure `DEREGISTER_CLUSTER` deregisters the Kafka cluster.

> **Note:**
>
> This procedure will only deregister and remove the cluster if there are no user applications associated with the cluster

Syntax

```
PROCEDURE DEREGISTER_CLUSTER (
                    cluster_name        IN VARCHAR2,
                    forced              IN BOOLEAN DEFAULT FALSE
  );
```

Parameters

Table 116-3 DEREGISTER_CLUSTER Procedure Parameters for DBMS_KAFKA_ADM

Parameter	Description
cluster_name	Name of an existing Kafka cluster Case-insensitive.
forced	(Optional) Marks the cluster as deregistered even if there are user applications still defined. Default: `FALSE`. Values: `[TRUE|FALSE]`

Usage Notes

When a cluster definition is no longer needed, the OSAK Administrator can remove the cluster definition.

If the `forced` parameter is passed as `TRUE`, then the cluster's state is marked as deregistered. Applications that are still associated with this cluster will no longer be functional. When the last application is dropped, this deregistered cluster definition will also be removed.

Examples

In the following example, an unused cluster definition for the Kafka cluster ExampleCluster is deregistered:

```
EXEC SYS.DBMS_KAFKA_ADM.DEREGISTER_CLUSTER ('ExampleCluster');
```

Suppose that there are still user applications (views and tables) associated with the cluster ExampleCluster, but you want to deregister the cluster. In the following example, the cluster ExampleCluster is deregistered with the forced parameter set to TRUE:

```
exec DBMS_KAFKA_ADM.DEREGISTER_CLUSTER(
    cluster_name => 'ExampleCluster',
    forced => TRUE);
```

DISABLE_CLUSTER

The `DBMS_KAFKA_ADM` procedure `DISABLE_CLUSTER` disables the Kafka cluster.

> **Note:**
>
> When this procedure completes successfully, the state of the application changes to `MAINTENANCE`.

Syntax

```
PROCEDURE DISABLE_CLUSTER(
        cluster_name        IN VARCHAR2
);
```

Parameters

Table 116-4 DISABLE_CLUSTER Procedure Parameters for DBMS_KAFKA_ADM

Parameter	Description
cluster_name	Name of an existing Kafka cluster Case-insensitive.

Usage Notes

A disabled cluster will prevent the cluster security information from being retrieved, thus preventing Kafka connections from being created or accessed.

Before you begin maintenance on a Kafka cluster, an Oracle SQL access to Kafka administrator should disable access to the cluster by using the `DBMS_KAFKA_ADM.DISABLE_CLUSTER` procedure. After the maintenance is completed, you can reeanble the cluster access by using the `DBMS_KAFKA_ADM.ENABLE_CLUSTER` procedure.

This procedure raises an exception if the state cannot be changed to `MAINTENANCE`

Examples

In the following example, the cluster definition for ExampleCluster is temporarily disabled. In this state, views cannot connect to or retrieve data from the Kafka cluster:

```
EXEC SYS.DBMS_KAFKA_ADM.DISABLE_CLUSTER ('ExampleCluster');
```

ENABLE_CLUSTER

The `DBMS_KAFKA_ADM` procedure `ENABLE_CLUSTER` enables the Kafka cluster.

> **Note:**
>
> When this procedure completes successfully, the state of the application changes to `CONNECTED`.

Syntax

```
PROCEDURE ENABLE_CLUSTER(
                           cluster_name    IN VARCHAR2
  );
```

Parameters

Table 116-5 ENABLE_CLUSTER Procedure Parameters for DBMS_KAFKA_ADM

Parameter	Description
cluster_name	Name of an existing Kafka cluster Case-insensitive.

Usage Notes

After you complete maintenance on a Kafka cluster, an Oracle SQL access to Kafka administrator should enable access to the cluster by using the `DBMS_KAFKA_ADM.ENABLE_CLUSTER` procedure.

This procedure raises an exception if the state cannot be changed to `CONNECTED`.

Examples

Suppose that you have disabled the cluster ExampleCluster to perform maintenance, and you now want to reenable the ability of views to connect to and retrieve data from that Kafka cluster. Enter the following command:

```
EXEC SYS.DBMS_KAFKA_ADM.ENABLE_CLUSTER ('ExampleCluster');
```

REGISTER_CLUSTER

The `DBMS_KAFKA_ADM` procedure `REGISTER_CLUSTER` registers the Kafka cluster and makes it available for access on Oracle Database.

Syntax

```
FUNCTION REGISTER_CLUSTER (
                      cluster_name             IN VARCHAR2,
                      startup_servers          IN VARCHAR2,
                      kafka_provider_provider  IN VARCHAR2,
                      cluster_access_dir       IN VARCHAR2,
                      credential_name          IN VARCHAR2 DEFAULT NULL,
                      cluster_config_dir       IN VARCHAR2 DEFAULT NULL,
                      cluster_description      IN VARCHAR2 DEFAULT NULL,
```

```
                        options                 IN CLOB DEFAULT NULL
)  RETURN INTEGER;
```

Parameters

Table 116-6 REGISTER_CLUSTER Procedure Parameters for DBMS_KAFKA_ADM

Parameter	Description
cluster_name	Name of an existing Kafka cluster Case-insensitive
bootstrap_servers	The startup servers of the Kafka cluster Case-sensitive
kafka_provider_*provider*	The provider of the Kafka server, which can be either Apache or Oracle Cloud Infrastructure (OCI) Oracle Streaming Service (OSS).. Syntax: DBMS_KAFKA_ADM.KAFKA_PROVIDER_*provider* Values for *provider*: [APACHE|OSS] Case-insensitive See also the DBMS_KAFKA_ADMconstant KAFKA_PROVIDER
cluster_access_dir	The Oracle directory object for determining access to this cluster Case-insensitive
credential_name	Credential name associated with the password to connect to Kafka. See SYS.DBMS_CREDENTIAL for possible name values and how to create a credential. Must be provided if Kafka connection requires a password, otherwise NULL
cluster_config_dir	The Oracle directory object containing the cluster configuration files required for secure clusters. The value is NULL for OSS or non- secure clusters. Case-insensitive
cluster_description	(Optional) A text description of the cluster. Maximum length: 400 characters.
options	(Optional) Cluster options. The format is a JSON-formatted document. For possible options, see SYS.DBMS_KAFKA_ADM.OPT_%

Usage Notes

REGISTER_CLUSTER automatically runs a CHECK_CLUSTER call after the cluster definition has been stored. The status returned from the CHECK_CLUSTER call is the returned value from the REGISTER_CLUSTER. This procedure raises an exception if the state cannot be determined.

At the end of the lifecycle for an Oracle SQL access to Kafka (OSAK) cluster, the administrator can remove the OSAK cluster by using the DBMS_KAFKA_ADM.DEREGISTER_CLUSTER function.

Examples

In the following example, the function request with REGISTER_CLUSTER registers the Kafka cluster KAFKACLUS1 that is on the server mykafkastartup, using Apache as the provider. The cluster access directory on the Oracle Database instance is OSAK_KAFKACLUS1_ACCESS,

accessed with the credential `KAFKACLUS1CRED1`, using the cluster configuration directory `OSAK_KAFKACLUS1_CONFIG`. The description for this cluster (which is optional), is "My test cluster kafkaclus1." A user (`examplekafka-user`) is granted `READ` access to the Kafka cluster data on `OSAK_KAFKACLUS1_ACCESS`,

```
SQL> select DBMS_KAFKA_ADM.REGISTER_CLUSTER ('KAFKACLUS1',
                                    'mykafkastartup-host:9092',
                                   DBMS_KAFKA_ADM.KAFKA_PROVIDER_APACHE,
                                   'OSAK_KAFKACLUS1_ACCESS'
                                   'KAFKACLUS1CRED1',
                                   'OSAK_KAFKACLUS1_CONFIG',
                                   'My test cluster kafkaclus1') from dual;

Output: Successful registration return 0 (zero)
SQL> DBMS_KAFKA_ADM_RE...
             0


        SQL> grant read on directory OSAK_KAFKACLUS1_ACCESS to examplekafka-
user;
```

You can also create non-secured cluster access. In the following example, a non-secure Kafka cluster access is registered using Oracle SQL access to Kafka (OSAK):

1. The cluster access database directory is created with an empty path. This directory is used to control which Oracle users can access the Kafka cluster:

   ```
   SQL> CREATE DIRECTORY
   OSAK_KAFKACLUS2_ACCESS AS '';
   ```

2. A cluster configuration operating system directory using the path structure *`Oracle base/`*osak/*cluster_name*/config is created, with the corresponding Oracle directory object, where the Oracle base is `/u01/app/oracle`, and the Kafka cluster name is `kafkaclus2`:

   ```
   $mkdir /u01/app/oracle/osak/kafkaclus2/config;
   .
   .
   .
   SQL> CREATE DIRECTORY OSAK_KAFKACLUS2_CONFIG AS 'u01/app/oracle/osak/
   kafkaclus2/config';
   ```

3. Create an empty `osakafka.properties` file, or create an `osakafka.properties` file with OSAK tuning or debugging properties, as you decide is required for your enterprise.
4. Register the Kafka cluster using `DBMS_KAKFA_ADM.REGISTER_CLUSTER()`. For example:

   ```
   SQL> select DBMS_KAFKA_ADM.REGISTER_CLUSTER (
             cluster_name  => 'KAFKACLUS2',
             bootstrap_servers =>'mykafkastartup-host:9092',
             kafka_provider => DBMS_KAFKA_ADM.KAFKA_PROVIDER_APACHE,
             cluster_access_dir => 'OSAK_KAFKACLUS2_ACCESS',
             credential_name => NULL,
             cluster_config_dir => 'OSAK_KAFKACLUS2_CONFIG',
              cluster_description =>  'My test cluster kafkaclus2',
   ```

```
            options  => NULL)
from dual;
Output: Successful registration return 0 (zero)
SQL> DBMS_KAFKA_ADM_RE…..
             0
```

5. Grant read access to users. In this example, user examplekafka-user2 is granted access:

```
SQL> grant read on directory
osak_kafkaclus2_access to examplekafka-user2;
```

UPDATE_CLUSTER_INFO

The `DBMS_KAFKA_ADM` procedure `UPDATE_CLUSTER_INFO` updates the Kafka host server, or the options for the Kafka cluster, or both.

Syntax

```
PROCEDURE UPDATE_CLUSTER_INFO(
                      cluster_name         IN VARCHAR2,
                      startup_servers    IN VARCHAR2 DEFAULT NULL,
                      options              IN CLOB DEFAULT NULL
 );
```

Parameters

Table 116-7 UPDATE_CLUSTER_INFO Procedure Parameters for DBMS_KAFKA_ADM

Parameter	Description
cluster_name	Name of an existing Kafka cluster Case-insensitive
startup_servers	The startup servers of the Kafka cluster Case-sensitive Because this is part of the Kafka source configuration, this string is treated as an opaque value (used as is), which is passed to the Kafka cluster connection logic.
options	(Optional) Cluster options. The format is a JSON-formatted document. For possible options, see `SYS.DBMS_KAFKA_ADM.OPT_%`

Usage Notes

The `UPDATE_CLUSTER_INFO` procedure updates the Kafka cluster definition, including the startup server list, or cluster options, or both. It also disconnects from the Kafka server. As the Oracle SQL access to Kafka (OSAK) Administrator, use this procedure if the Kafka cluster environment changes, and you need to change the cluster definition and configuration.

If an update fails, then the procedure raises an exception.

Examples

In the following example, the Kafka administrator has reconfigured the cluster to use the servers `newhost` and `host2`, and is now updating the list of startup servers for the Kafka cluster `ExampleCluster`:

```
EXEC SYS.DBMS_KAFKA_ADM.UPDATE_CLUSTER_INFO
('ExampleCluster',

'newhost:9092,host2:9092');
```