DataStax OBDC driver

Documentation

${ds.localized.time}

About the driver

The DataStax ODBC driver for Apache Cassandra and DataStax Enterprise with CQL connector enables *Business Intelligence* (BI), analytics, and reporting on data that is stored in Apache Cassandra databases. The driver complies with the ODBC 3.52 data standard and adds important functionality such as Unicode, as well as 32- and 64-bit support for high-performance computing environments on all platforms.

ODBC is one the most established and widely supported APIs for connecting to and working with databases. At the heart of the technology is the ODBC driver, which connects an application to the database.

The driver was developed by Simba Corporation and is distributed for free by DataStax for all users of Apache Cassandra and DataStax Enterprise.

Features

Query modes

SQL with CQL fallback

SQL with CQL fallback is the default query mode used by the driver. In this query mode, the driver attempts to treat the incoming query as SQL first. If an error occurs while handling the query as SQL, then the driver passes the original query to Cassandra to execute as CQL.

Note: The SQL with CQL fallback query mode will not work if the incoming query references virtual tables, as Cassandra is not aware of virtual tables. When querying against virtual tables, ensure that the query syntax is valid SQL-92 syntax.

CQL

In this query mode, the driver treats all incoming queries as CQL and does not accept any non- CQL syntax.

SQL

In this query mode, the driver treats all incoming queries as SQL and does not accept any CQL query syntax that is not standard SQL-92 syntax.

Authentication

The Cassandra service supports authentication through user login. Configure authentication for your connection by enabling the authentication mechanism and then specifying a user name and password in the DSN or in the connection string. You can also use the features available in your client application to implement access control.

Currently the driver does not support SSL connections.

Catalog and schema support

The driver supports both catalogs and schemas in order to make it easy to work with various ODBC applications. Since Cassandra only organizes column families into keyspaces, the driver adds a synthetic catalog called CASSANDRA under which all keyspaces are organized.

Data types

The driver supports all the Cassandra data types.

The following data types are supported:

AsciiType	FloatType	TimestampType
BigintType	InetType	UuidType
BlobType	IntType	TimeuuidType
BooleanType	ListType	VarcharType
CounterType	MapType	VarintType

`DecimalType`	`SetType`	
`DoubleType`	`TextType`	

Note: Cassandra internally represents Timestamp value as a 64-bit signed integer value representing the number of milliseconds since epoch “January 1 1970 at 00:00:00 GMT”. The range of Timestamp values supported by the DataStax ODBC driver for Apache Cassandra and DataStax Enterprise with CQL connector is from “1601-01-01 00:00:00.000” to “9999-12-31 23:59:59.999”.

To support complex data types such as sets, lists, and maps, the driver converts the data into virtual tables.

Virtual tables

A Cassandra collection data type is stored in a virtual table simplifying schema management.

One advantage of the Apache Cassandra design is the ability to store data that is denormalized into a fewer number of tables. By taking advantage of nested data structures such as collections (that is `sets`, `lists`, and `maps`), transactions can be simplified. However, the ODBC interface does not natively support accessing this type of data. By expanding the data contained within collections into virtual tables, the DataStax ODBC driver for Apache Cassandra and DataStax Enterprise with CQL connector allows users to directly interact with the data but leave the storage of the data in its denormalized form in Cassandra.

If a table contains any collection columns, when the table is queried for the first time, the driver creates the following virtual tables and saves them as part of the schema definition:

- A *main* virtual table, which contains the same data as the real table except for the collection columns.
- A virtual table for each collection column, expanding the first level of nested data.

Virtual tables refer to the data in the real table, enabling the driver to access the denormalized data. By querying the virtual tables, you can access the contents of Cassandra collections via ODBC. When you write or modify data in a virtual table, the data in the real table in the Cassandra database is updated.

Virtual tables appear as additional tables in the list of tables that exist in the database. The main virtual table uses the same name as the real table that it represents. The virtual tables that represent collections are named using the name of the real table, a separator (_vt_), and the name of the column.

For example, consider the following table in Cassandra called `ExampleTable`, which contains an integer primary key column named `pk_int`, a list column, a map column, and a set column (named `StringSet`):

Table 1: ExampleTable

pk_int	List	Map	StringSet
1	[“1”, “2” , “3”]	{“S1”:“a”, “S2”:“b”}	{ “A”, “B”, “C” }
3	[“100”, “101”, “102”, “105”]	{ “S1” : “t” }	{ “A”, “E” }

The driver would generate multiple virtual tables to represent this single table. The first virtual table is the main virtual table:

Table 2: ExampleTable

pk_int
1
3

The main virtual table contains all of the data of the original table, but the data from the collections have been omitted and will be expanded in the other virtual tables.

The following are the virtual tables that represent the data from the List, Map, and StringSet columns:

Table 3: ExampleTable_vt_List

pk_int	List#index	List#vlaue
1	0	1
1	1	2
1	2	3
3	0	100
3	1	101
3	2	102
3	3	105

Table 4: ExampleTable_vt_Map

pk_int	Map#index	Map#vlaue
1	S1	a
1	S2	b
3	S1	t

Table 5: ExampleTable_vt_StringSet

pk_int	StringSet#value
1	A
1	B
1	C
3	A
3	E

The foreign key columns in the virtual tables reference the primary key columns in the real table, and indicate which real table row the virtual table row corresponds to. The columns with names that end with `#index` or `#key` indicate the position of the data within the original list, map, or set. The columns with names that end with `#value` contain the expanded data from the collection.

The data in the virtual tables can be selected, inserted, and updated as if they were normal tables, and the driver will handle the storage details within Cassandra. You can also explicitly append data to the end of a list by inserting a row of data with the index column set to -1.

For example, to append 106 to the List column in ExampleTable, where `pk_int = 3`, you would run the following query:

```
INSERT INTO “ExampleTable_vt_List” (pk_int, “List#index”, “List#value”)
 VALUES (3, -1, ‘106’)
```

Write-back

The driver supports Data Manipulation Languages (DML) statements such as `INSERT`, `UPDATE`, and `DELETE`.

Note: Because Cassandra only provides the `UPSERT` operation instead of `INSERT` and `UPDATE`, when you execute an `INSERT` or `UPDATE` statement using the driver, it will behave like an `UPSERT` operation. This means that both `INSERT` and `UPDATE` will set the column value regardless of whether the data already exists.

TRUNCATE TABLE

TRUNCATE TABLE syntax is neither SQL or CQL.

The `TRUNCATE TABLE tableName` syntax is neither standard SQL-92 nor CQL. The driver translates the `TRUNCATE TABLE tableName` syntax into `TRUNCATE tableName` CQL syntax for non-virtual tables. Use `DELETE FROM virtualTableName` to delete all the rows from a virtual table.

SSL support

- Windows
- Linux

Windows driver

Installing the Windows Driver

Installing the Windows 32- or 64-bit version of the driver.

About this task

On 64-bit Windows operating systems, you can execute 32- and 64-bit applications transparently. You must use the version of the driver matching the bitness of the client application accessing Cassandra databases:

- `DataStaxCassandraODBC32.msi` for 32-bit applications
- `DataStaxCassandraODBC64.msi` for 64-bit applications

  You can install both versions of the driver on the same computer.

Note: How to use ODBC on 64-bit editions of Windows.

Before you begin

You install the driver on client computers accessing data in Cassandra databases. Each computer where you install the driver must meet the following minimum system requirements:

- One of the following operating systems (32- and 64-bit editions are supported):
  - Windows® XP with SP3
  - Windows® Vista
  - Windows® 7 Professional
  - Windows® Server 2008 R2
- 30 MB of available disk space.

  The driver supports Apache Cassandra 2.0 or later.

  Important: To install the driver, you need Administrator privileges on the computer.

Procedure

1. Depending on the bitness of your client application, double-click to run `DataStaxCassandraODBC32.msi` or `DataStaxCassandraODBC64.msi`.
2. Click **Next**.
3. Select the check box to accept the terms of the License Agreement (if you agree) and then click **Next**.
4. To change the installation location, click **Change**, then browse to the desired folder, and then click **OK**. To accept the installation location, click **Next**.
5. Click **Install**.
6. When the installation completes, click **Finish**.

Configuring a Data Source Name (DSN)

How to configure a DSN.

About this task

After installing the driver, you need to create a Data Source Name (DSN).

Procedure

1. Click **Start**, then click **All Programs**, then click the **DataStax ODBC Driver 1.0** program group corresponding to the bitness of the client application accessing data in Cassandra, and then click **ODBC Administrator**.
2. In the ODBC Administrator, click the **Drivers** tab and then scroll down as needed to verify that the DataStax Cassandra ODBC Driver appears in the alphabetical list of driver names.
3. Create a DSN:
   - To create a DSN on the computer that only the user currently logged into Windows can use, click the **User DSN** tab.
   - To create a DSN on the computer that all users who log into Windows can use, click the **System DSN** tab.
4. Click **Add**.
5. In the **Create New Data Source** dialog box, select **DataStax Cassandra ODBC Driver** and then click **Finish**.
6. Use the options in the **DataStax Cassandra ODBC Driver DSN Setup** dialog box to configure your DSN:
   a) In the **Data Source Name** field, type a name for the data source.
   b) Optionally, in the **Description** field, type relevant details about the DSN.
   c) In the **Host** field, do one of the following:
      - Type the name or IP address of the host where your Cassandra instance is running.
      - Type a comma-separated list of hostnames or IP addresses of Cassandra servers to which the driver connects.

      Note: The driver attempts to connect to all the servers concurrently and then keep the first connection that is successfully established. The driver does not maintain a connection with any of the other servers in the list.
   d) In the **Port** field, type the number of the port that the Cassandra instance uses.

      The default port used by Cassandra is 9042.
   e) If user login is required to access the Cassandra instance, then do the following:
      1. In the **Mechanism** list, select **User Name** and **Password**
      2. In the **User Name** field, type an appropriate user name for accessing the Cassandra instance.
      3. In the **Password** field, type the password corresponding to the user name you typed in the previous step.
   f) In the **Default Keyspace** field, type the name of the Cassandra keyspace to use by default.
   g) To configure advanced driver options, click **Advanced Options**.
   h) To set llogging behavior for the driver, click **Logging Options**.
7. To confirm that the DSN connects to your Cassandra database, click **Test**. Review the results as needed and then click **OK**.
8. To save your settings and close the **DataStax Cassandra ODBC Driver DSN Setup** dialog box, click **OK**.
9. To close the **ODBC Data Source Administrator**, click **OK**.

Configuring Advanced Options

About this task

The driver has advanced configuration options.

You configure advanced options using the following:

- data source name
- database connection string

Procedure

To set advanced options using the **DataStax Cassandra ODBC Driver DSN Setup** dialog box:

1. In the **ODBC Data Source Administrator** where you created the DSN, select the **DSN** tab where the Data Source Name appears, and then select the **Data Source Name**.
2. Click **Configure** and then click **Advanced Options**.
3. In the **Query Mode** list, select an option to specify how the driver executes queries:
   - To execute all queries in SQL, select SQL.
   - To execute all queries in CQL, select CQL.
   - To execute queries in SQL by default and then execute failed queries in CQL, select SQL with CQL fallback.
4. In the **Tunable consistency** list, select an option to specify a Cassandra replica or the number of Cassandra replicas that must process a query in order for the query to be considered successful.
5. In the **Binary column** length field, type the default column length to report for `blob` columns.
6. In the **String column** length field, type the default column length to report for `ascii`, `text`, and `varchar` columns.
7. In the **Virtual table name** separator field, type a separator for naming a virtual table built from a collection.
8. To map `text` and `varchar` data types in Cassandra to use `SQL_WVARCHAR`, select the `Use SQL_WVARCHAR for string data type` check box.
9. Configure the results:
   - To configure the driver to split large result sets into pages, select the **Enable paging** check box and then type the maximum number of rows to display on each page in the Rows per page field.
   - To configure the driver to fetch all results into memory regardless of the result set size, clear the **Enable paging** check box.

     Important: Disabling paging and then fetching a large result set can cause issues such as out of memory errors and database timeouts.
10. To save your settings and close the **Advanced Options** dialog box, click **OK**.
11. To close the **DataStax Cassandra ODBC Driver DSN Setup** dialog box, click **OK**.

Using a database connection string

Here is an example database connection string that sets advanced options:

```
DSN=DataStax Cassandra ODBC DSN;
Host=server1,server2,server3;
Port=9042;
DefaultKeyspace=MyKeyspace;
StringColumnLength=4000;
BinaryColumnLength=4000;
```

```
QueryMode=0;
```

server1, *server2*, and *server3* are the hostnames or IP addresses of the Cassandra servers to which the driver connects. *MyKeyspace* is the Cassandra keyspace to use.

Configuring SSL verification

You can configure verification between the client and the Cassandra server over SSL.

Configuring a connection without SSL

About this task

To configure a connection without SSL:

Procedure

1. Access the SSL options for a DSN:
   a) Open the **ODBC Data Source Administrator** where you created the DSN.
   b) Select the DSN, then click **Configure**.
   c) Click **Advanced Options**.
   d) In the SSL area, select **No Verification**.
2. Save your settings and close the dialog box, by clicking **OK**.

Configuring one-way SSL verification

About this task

You can enable the client to verify the Cassandra server, by configuring one-way SSL verification:

Procedure

1. Access the SSL options for a DSN:
   a) Open the **ODBC Data Source Administrator** where you created the DSN.
   b) Select the DSN.
   c) Click **Configure**,
   d) Click **Advanced Options**.
   e) In the **SSL** area, select **One-way Server Verification**.
   f) In the **Trusted CA Certificates** field, specify the full path of the PEM file containing the certificate for verifying the server.
   g) (Optional), configure the driver to require the host name of the server to match the host name in the certificate by selecting the **Enable Server Hostname Verification** check box.
2. Save your settings by closing the dialog box: click **OK**.

Configuring two-way SSL verification

About this task

You can enable the client and the Cassandra server to verify each other. To configure two-way SSL verification:

Procedure

1. Access the SSL options for a DSN:

a) Open the **ODBC Data Source Administrator** where you created the DSN,
b) Select the DSN.
c) Click **Configure**,
d) Click **Advanced Options**.
e) In the **SSL** area, select **Two-way Server and Client Verification**.
f) In the **Trusted CA Certificates** field, specify the full path of the PEM file containing the certificate for verifying the server.
g) In the **Client-side Certificate** field, specify the full path of the PEM file containing the certificate for verifying the client.
h) In the **Client-side Private Key** field, specify the full path of the file containing the private key used to verify the client.
i) If the private key file is protected with a password, type the password in the **Key File Password** field. To save the password in the DSN, select the Remember Password check box. Passwords are saved in plain text in the DSN; they are not encrypted or censored.
j) (Optional) configure the driver to require the host name of the server to match the host name in the certificate by selecting the **Enable Server Hostname Verification** check box.

2. Save your settings by closing the dialog box: click **OK**.

Configuring Logging Options

To help troubleshoot issues, you can enable logging. In addition to functionality provided in the DataStax ODBC driver for Apache Cassandra and DataStax Enterprise with CQL connector, the ODBC Data Source Administrator provides tracing functionality.

Important: Only enable logging long enough to capture an issue. Logging decreases performance and can consume a large quantity of disk space.

The driver allows you to set the amount of detail included in log files. Table 1 lists the logging levels provided by the DataStax ODBC driver for Apache Cassandra and DataStax Enterprise with CQL connector, in order from least verbose to most verbose.

Table 6: DataStax ODBC driver for Apache Cassandra and DataStax Enterprise with CQL connector Logging Levels

Logging Level	Description
OFF	Disables all logging.
FATAL	Logs very severe error events that will lead the driver to abort.
ERROR	Logs error events that might still allow the driver to continue running.
WARNING	Logs potentially harmful situations.
INFO	Logs general information that describes the progress of the driver.
DEBUG	Logs detailed information that is useful for debugging the driver.
TRACE	Logs more detailed information than the DEBUG level.

Enabling driver logging

Procedure

1. In the **DataStax Cassandra ODBC Driver DSN Setup** dialog, click **Logging Options**.
2. In the **Log Level** list, select the desired level of information to include in log files.

3. In the **Log Path** field, type the full path to the folder where you want to save log files.
4. If requested by DataStax Technical Support, type the name of the component for which to log messages in the **Log Namespace** field. Otherwise, do not type a value in the field.
5. Click **OK**.
   The driver produces a log file named `DataStax Cassandra ODBC Driver_driver.log` at the location you specify using the **Log Path** field.

Disabling driver logging

Procedure

1. In the **DataStax Cassandra ODBC Driver DSN Setup** dialog, click **Logging Options**.
2. In the **Log Level** list, select **LOG_OFF**.
3. Click **OK**.

Start tracing

Procedure

1. In the **ODBC Data Source Administrator**, click the **Tracing** tab.
2. In the **Log File Path** area, click **Browse**.
3. In the **Select ODBC Log File** dialog, browse to the location where you want to save the log file, then type a descriptive file name in the **File name** field, and then click **Save**.
4. On the **Tracing** tab, click **Start Tracing Now**

Stopping tracing

Procedure

1. In the **ODBC Data Source Administrator**, select the **Tracing** tab.
2. Click **Stop Tracing Now**.

Linux Driver

Installing the Linux driver

Installing the driver on Linux using an RPM.

Before you begin

You install the driver on client computers accessing data in Cassandra databases. Each computer where you install the driver must meet the following minimum system requirements:

- One of the following operating systems (32- and 64-bit editions are supported):
  - Red Hat® Enterprise Linux® (RHEL) 5.x or 6.x
  - CentOS5.x or 6.x
  - SUSE Linux Enterprise Server (SLES) 11
- 45 MB of available disk space
- One of the following ODBC driver managers installed:
  - iODBC 3.52.7 or above
  - unixODBC 2.2.12 or above

The driver supports Apache Cassandra 2.0 or later.

About this task

There are two versions of the driver for Linux:

- `DataStaxCassandraODBC-32bit-Version.rpm` for 32-bit
- `DataStaxCassandraODBC-64bit-Version.rpm` for 64-bit

  The version of the driver that you select should match the bitness of the client application accessing your Cassandra data. For example, if the client application is 64-bit, then you should install the 64-bit driver. Note that 64-bit editions of Linux support both 32- and 64-bit applications. Verify the bitness of your intended application and install the appropriate version of the driver.

  Important: Ensure that you install the driver using the RPM corresponding to your Linux distribution.

The driver files are installed in the following directories:

- `/opt/datastax/cassandraodbc` contains release notes and the DataStax ODBC driver documentation in PDF format.
- `/opt/datastax/cassandraodbc/lib/32` contains the 32-bit driver and the `datastax.cassandraodbc.ini` configuration file.
- `/opt/datastax/cassandraodbc/lib/64` contains the 64-bit driver and the `datastax.cassandraodbc.ini` configuration file.
- `/opt/datastax/cassandraodbc/ErrorMessages` contains error message files required by the driver.
- `/opt/datastax/cassandraodbc/Setup` contains sample configuration files named `odbc.ini` and `odbcinst.ini`

- In Red Hat Enterprise Linux or CentOS, log in as the root user, then navigate to the folder containing the driver RPM packages to install, and then type the following at the command line, where

RPMFileName is the file name of the RPM package containing the version of the driver that you want to install.

```
$ yum --nogpgcheck localinstall  RPMFileName
```

- In SUSE Linux Enterprise Server, log in as the root user, then navigate to the folder containing the driver RPM packages to install, and then type the following at the command line, where *RPMFileName* is the file name of the RPM package containing the version of the driver that you want to install.

```
$ zypper install RPMFileName
```

Setting the LD_LIBRARY_PATH environment variable

Letting the driver know where its manager libraries are installed.

About this task

The `LD_LIBRARY_PATH` environment variable must include the paths to the installed ODBC driver manager libraries.

Procedure

For example, if ODBC driver manager libraries are installed in `/usr/local/lib`, then set `LD_LIBRARY_PATH` as follows:

```
$ export LD_LIBRARY_PATH=/usr/local/lib
```

For information about how to set environment variables permanently, refer to your Linux shell documentation.

Configuring ODBC Connections

ODBC driver managers use configuration files to define and configure ODBC data sources and drivers. By default, the following configuration files residing in the user's home directory are used:

- `.odbc.ini` is used to define ODBC data sources, and it is required.
- `.odbcinst.ini` is used to define ODBC drivers, and it is optional.

Also, by default the driver is configured using the `datastax.cassandraodbc.ini` file in the subfolder in the `/lib` folder containing the binaries for the driver that you are using. The `datastax.cassandraodbc.ini` file is required.

You can set driver configuration options in your `odbc.ini` and `datastax.cassandraodbc.ini` files. Configuration options set in a `datastax.cassandraodbc.ini` file apply to all connections, whereas configuration options set in an `odbc.ini` file are specific to a connection. Configuration options set in `odbc.ini` take precedence over configuration options set in `datastax.cassandraodbc.ini`.

Sample files

The driver installation contains the following sample configuration files in the `Setup` directory:

- `odbc.ini`
- `odbcinst.ini`

The sample configuration files in the `Setup` directory provide preset values for settings related to the driver.

The names of the sample configuration files do not begin with a period (.) so that they will appear in directory listings by default. A filename beginning with a period (.) is hidden. For `odbc.ini` and `odbcinst.ini`, if the default location is used, then the filenames must begin with a period (.).

If the configuration files do not already exist in the user's home directory, then you can copy the sample configuration files to that directory and rename them. If the configuration files already exist in the home directory, then use the sample configuration files as a guide to modify the existing configuration files.

Configuring the environment

Optionally, you can use three environment variables—`ODBCINI`, `ODBCSYSINI`, and `DATASTAXCASSANDRAODBC`—to specify different locations for the `odbc.ini`, `odbcinst.ini`, and `datastax.cassandraodbc.ini` configuration files by doing the following:

- Set `ODBCINI` to point to your `odbc.ini` file.
- Set `ODBCSYSINI` to point to the directory containing the `odbcinst.ini` file.
- Set `DATASTAXCASSANDRAODBC` to point to your `datastax.cassandraodbc.ini` file.

  For example, if your `odbc.ini` and `datastax.cassandraodbc.ini` files are located in the folder `/etc` and your `odbcinst.ini` file is located in the folder `/usr/local/odbc`, then set the environment variables as follows:

```
export ODBCINI=/etc/odbc.ini
export ODBCSYSINI=/usr/local/odbc
export DATASTAXCASSANDRAODBC=/etc/datastax.cassandraodbc.ini
```

  The following search order is used to locate the `datastax.cassandraodbc.ini` file:

  1. If the `DATASTAXCASSANDRAODBC` environment variable is defined, then the driver searches for the file specified by the environment variable.

     Important: `DATASTAXCASSANDRAODBC` must contain the full path, including the filename.
  2. The current working directory of the application is searched for a file named `datastax.cassandraodbc.ini` not beginning with a period.
  3. The directory `~/` (that is, `$HOME`) is searched for a hidden file named `.datastax.cassandraodbc.ini`.
  4. The directory /etc is searched for a file named `datastax.cassandraodbc.ini` not beginning with a period.

Configuring the odbc.ini file

ODBC Data Source Names (DSNs) are defined in the odbc.ini configuration file. The file is divided into several sections:

- `[ODBC]` is optional and used to control global ODBC configuration, such as ODBC tracing.
- `[ODBC Data Sources]` is required, listing DSNs and associating DSNs with a driver.
- A section having the same name as the data source specified in the `[ODBC Data Sources]` section is required to configure the data source.

The following is an example of an odbc.ini configuration file:

```
[ODBC Data Sources]
DataStax Cassandra ODBC DSN 32=DataStax Cassandra ODBC Driver 32-bit
[DataStax Cassandra ODBC DSN 32]
Driver=/opt/datastax/cassandraodbc/lib/32/libdatastaxcqlodbc32.so
HOST=server1,server2,server3
PORT=9042
QueryMode=0
```

```
DefaultKeyspace=MyKeyspace
```

server1, *server2*, and *server3* are the hostnames or IP addresses of the Cassandra servers to which the driver connects. *MyKeyspace* is the Cassandra keyspace to use.

Creating a Data Source Name (DSN)

About this task

There are configuration options available for controlling the behavior of DSNs that are using the driver.

Procedure

1. Open the `.odbc.ini` configuration file in a text editor.
2. In the `[ODBC Data Sources]` section, add a new entry by typing the Data Source Name (DSN), then an equal sign (=), and then the driver name.

```
[ODBC Data Sources]
Data Source Name=driver-name
```

3. In the `odbc.ini` file, add a new section with a name that matches the DSN you specified in the previous step, and then add configuration options to the section. Specify the configuration options as key-value pairs.
4. Save the `.odbc.ini` configuration file.

Configuring the odbcinst.ini file

ODBC Drivers are defined in the odbcinst.ini configuration file. The configuration file is optional because drivers can be specified directly in the odbc.ini configuration file, as described in “Configuring the odbc.ini File” on page 14.

The odbcinst.ini file is divided into the following sections:

- [ODBC Drivers] lists the names of all the installed ODBC drivers.
- A section having the same name as the driver name specified in the [ODBC Drivers] section lists driver attributes and values.

The following is an example of an odbcinst.ini configuration file:

```
[ODBC Drivers]
DataStax Cassandra ODBC Driver 32-bit=Installed
DataStax Cassandra ODBC Driver 64-bit=Installed
DataStax Cassandra ODBC Driver 32-bit]
Description=DataStax Cassandra ODBC Driver (32-bit)
Driver=/opt/datastax/cassandraodbc/lib/32/libdatastaxcqlodbc32.so [
DataStax Cassandra ODBC Driver 64-bit]
Description=DataStax Cassandra ODBC Driver (64-bit)
Driver=/opt/datastax/cassandraodbc/lib/64/libdatastaxcqlodbc64.so
```

Defining a driver

Procedure

1. Open the `.odbcinst.ini` configuration file in a text editor.
2. In the `[ODBC Drivers]` section, add a new entry by typing the driver name and then typing:

```
[ODBC Drivers]
```

```
driver-name=Installed
```

Note: Type a symbolic name that you want to use to refer to the driver in connection strings or DSNs.

3. In the `.odbcinst.ini` file, add a new section with a name that matches the driver name you typed in the previous step, and then add configuration options to the section based on the sample `odbcinst.ini` file provided in the `Setup` directory. Specify the configuration options as key-value pairs.
4. Save the `.odbcinst.ini` configuration file.

Configuring the datastax.cassandraodbc.ini file

About this task

To configure the driver to work with your ODBC driver manager:

Procedure

1. Open the `datastax.cassandraodbc.ini` configuration file in a text editor.
2. Edit the `DriverManagerEncoding` setting. The value is usually UTF-16 or UTF-32, depending on the ODBC driver manager you use. iODBC uses UTF-32, and unixODBC uses UTF-16. To determine the correct setting to use, refer to your ODBC Driver Manager documentation.
3. Edit the `ODBCInstLib` setting. The value is the name of the ODBCInst shared library for the ODBC driver manager you use. To determine the correct library to specify, refer to your ODBC driver manager documentation.
   The configuration file defaults to the shared library for iODBC. In Linux, the shared library name for iODBC is libiodbcinst.so.

   Note: You can specify an absolute or relative filename for the library. If you intend to use the relative filename, then the path to the library must be included in the library path environment variable. In Linux, the library path environment variable is named `LD_LIBRARY_PATH`.
4. Optionally, configure logging by editing the `LogLevel` and `LogPath` settings.
5. Save the `datastax.cassandraodbc.ini` configuration file.

Configuring SSL verification

You can configure verification between the client and the Cassandra server over SSL by defining the SSLMode connection attribute in a connection string or in a DSN (in the `odbc.ini` file). Depending on the type of SSL verification that you use, there may be additional connection attributes that you must define. For more information about the attributes involved in configuring SSL verification, see Driver configuration options .

Configuring a connection without SSL

You can disable SSL verification between the client and the server if it is not needed.

About this task

To configure a connection without SSL:

Procedure

Set the SSLMode connection attribute to 0.

Configuring one-way SSL verification

About this task

You can enable the client to verify the Cassandra server, by configuring one-way SSL verification:

Procedure

1. Set the SSLMode connection attribute to 1.
2. Set the SSLTrustedCertsPath connection attribute to the full path of the PEM file containing the certificate for verifying the server.
3. (Optional) configure the driver to require the host name of the server to match the host name in the certificate by setting the UseSslIdentityCheck connection attribute to 1.

Configuring two-way SSL verification

You can enable the client and the Cassandra server to verify each other.

About this task

To configure two-way SSL verification:

Procedure

1. Set the SSLMode connection attribute to 2.
2. Set the SSLTrustedCertsPath connection attribute to the full path of the PEM file containing the certificate for verifying the server.
3. Set the SSLUserCertsPath connection attribute to the full path of the PEM file containing the certificate for verifying the client.
4. Set the SSLUserKeyPath connection attribute to the full path of the file containing the private key used to verify the client.
5. If the private key file is protected with a password, set the SSLUserKeyPWD connection attribute to specify the password. Passwords are saved in plain text in the DSN; they are not encrypted or censored.
6. (Optional) configure the driver to require the host name of the server to match the host name in the certificate by setting the UseSslIdentityCheck connection attribute to 1.

Configuring logging options

To help troubleshoot issues, you can enable logging in the driver.

Important: Only enable logging long enough to capture an issue. Logging decreases performance and can consume a large quantity of disk space.

Use the LogLevel key to set the amount of detail included in log files. Table 2 lists the logging levels provided by the DataStax ODBC driver for Apache Cassandra and DataStax Enterprise with CQL connector, in order from least verbose to most verbose.

Table 7: DataStax ODBC driver for Apache Cassandra and DataStax Enterprise with CQL connector Logging Levels

Logging Level	Description
0	Disables all logging.
1	Logs very severe error events that will lead the driver to abort.
2	Logs error events that might still allow the driver to continue running.

Logging Level	Description
3	Logs potentially harmful situations.
4	Logs general information that describes the progress of the driver.
5	Logs detailed information that is useful for debugging the driver.
6	Logs more detailed information than the LogLevel=5.

Disabling logging

Procedure

1. Open the datastax.cassandraodbc.ini configuration file in a text editor.
2. Set the LogLevel key to 0
3. Save the datastax.cassandraodbc.ini configuration file.

Driver configuration options

The Configuration options table lists and describes the configuration options that are available in the DataStax ODBC driver for Apache Cassandra and DataStax Enterprise with CQL connector by field or button label.

When creating or configuring a DSN connection from a Windows machine, the fields and buttons described in Table 3 are available in the DataStax Cassandra ODBC Driver DSN Setup dialog box and the Advanced Options dialog box.

When using a connection string or configuring a connection from a Linux machine, use the key names provided in Table 3.

Note: You can set driver configuration options in your odbc.ini and datastax.cassandraodbc.ini files. Configuration options set in a datastax.cassandraodbc.ini file apply to all connections, whereas configuration options set in an odbc.ini file are specific to a connection. Configuration options set in odbc.ini take precedence over configuration options set in datastax.cassandraodbc.ini.

Table 8: Configuration options

Field Label (Key)	Default Value	Description
Binary Column Length (BinaryColumnLength)	4000	The default column length to report for BLOB columns. (Optional)
Default Keyspace (DefaultKeyspace)	None	The default keyspace (schema) to connect to in Cassandra. (Optional)
Enable Paging (EnablePaging)	1	Whether the driver should split large result sets into pages. The following values are possible: • 0—Fetch all results into memory regardless of the result set size. • 1—Split result sets into pages. See also the Rows Per Page driver configuration option. (Optional)
Host (Host)	None	The hostname or IP address of the Cassandra server to which the driver connects. You can specify a comma-separated list of hostnames or IP addresses. The driver will attempt to connect to all the servers concurrently, and then keep the first connection that is successfully established. The driver does not maintain a connection with any of the other servers in the list. (Required)
Mechanism (AuthMech)	O	The authentication mechanism to use. The following values are possible: • 0—No Authentication • 1—Username and Password (Optional)
Password (PWD)	None	The password corresponding to the user name that you provided in the User Name field (the UID key).

Field Label (Key)	Default Value	Description
		(Required if the authentication mechanism is User Name and Password)
Port (Port)	9042	The number of the port that the Cassandra server uses. (Optional)
Query Mode (QueryMode)	2	The query mode to use when sending queries to Cassandra. The following values are possible: • 0—The driver uses SQL_QUERY_MODE and executes all queries in SQL. • 1—The driver uses CQL_QUERY_MODE and executes all queries in CQL. • 2—The driver uses SQL_WITH_CQL_FALLBACK_QUERY_MO DE and executes all queries in SQL by default. If a query fails, then the driver executes the query in CQL. (Optional)
Rows Per Page (RowsPerPage)	10000	When result set paging is enabled, use this option to specify the maximum number of rows to display on each page. See also the Enable Paging driver configuration option. (Optional)
String Column Length (StringColumnLength)	4000	The default column length to report for ASCII, TEXT, and VARCHAR columns. (Optional)
Tunable Consistency (TunableConsistency)	1	he specific Cassandra replica or the number of Cassandra replicas that must process a query in order for the query to be considered successful. The following values are possible, and each value corresponds to one of the consistency levels available in Cassandra: • 0—The ANY consistency level. • 1—The ONE consistency level. • 2—The TWO consistency level. • 3—The THREE consistency level. • 4—The QUORUM consistency level. • 5—The ALL consistency level. • 6—The LOCAL_QUORUM consistency level. • 7—The EACH_QUORUM consistency level. • 10—The LOCAL_ONE consistency level. **1.** For detailed information about each consistency level, see “Configuring data consistency” in the Apache Cassandra 2.0 documentation: http://www.datastax.com/en/cassandra/2.0/cassandra/dml/dml_config_consistency_ c.html (Optional)
Use SQL_WVARCHAR for string data types (UseSqlWVarchar)	0	Whether the Cassandra data types text and varchar should be mapped to SQL_VARCHAR

Field Label (Key)	Default Value	Description
		or SQL_WVARCHAR. The following values are possible: • 0—The Cassandra data types text and varchar are mapped to SQL_VARCHAR. • 1—The Cassandra data types text and varchar are mapped to SQL_WVARCHAR. (Optional)
User Name (UID)	None	The user name that you use to access the Cassandra server. (Required if the authentication mechanism is User Name and Password)
Virtual table name separator (VTTableNameSeparator)	_vt_	The separator for naming a virtual table built from a collection. The name of a virtual table consists of the name of the original table, then the separator, and then the name of the collection. For example, OriginalTable_vt_CollectionName