Magnitude Simba Cassandra ODBC Data Connector

Installation and Configuration Guide

Version 2.7.3

October 2023

Copyright

Contact Us

Magnitude Software, Inc.

www.magnitude.com

About This Guide

Purpose

The *Magnitude Simba Cassandra ODBC Data Connector Installation and Configuration Guide* explains how to install and configure the Magnitude Simba Cassandra ODBC Data Connector. The guide also provides details related to features of the connector.

Audience

The guide is intended for end users of the Simba Cassandra ODBC Connector, as well as administrators and developers integrating the connector.

Knowledge Prerequisites

To use the Simba Cassandra ODBC Connector, the following knowledge is helpful:

- Familiarity with the platform on which you are using the Simba Cassandra ODBC Connector
- Ability to use the data source to which the Simba Cassandra ODBC Connector is connecting
- An understanding of the role of ODBC technologies and driver managers in connecting to a data source
- Experience creating and configuring ODBC connections
- Exposure to SQL

Document Conventions

Italics are used when referring to book and document titles.

Bold is used in procedures for graphical user interface elements that a user clicks and text that a user types.

`Monospace font` indicates commands, source code, or contents of text files.

> **Note:**
>
> A text box with a pencil icon indicates a short note appended to a paragraph.

Important:

A text box with an exclamation mark indicates an important comment related to the preceding paragraph.

Contents

About the Simba Cassandra ODBC Connector

The Simba Cassandra ODBC Connector enables Business Intelligence (BI), analytics, and reporting on data that is stored in Apache Cassandra databases, including support for databases hosted on the cloud as DataStax Astra databases. The connector complies with the ODBC 3.80 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 of the most established and widely supported APIs for connecting to and working with databases. At the heart of the technology is the ODBC connector, which connects an application to the database. For more information about ODBC, see *Data Access Standards* on the Simba Technologies website: https://www.simba.com/resources/data-access-standards-glossary. For complete information about the ODBC specification, see the *ODBC API Reference* from the Microsoft documentation: https://docs.microsoft.com/en-us/sql/odbc/reference/syntax/odbc-api-reference.

The Simba Cassandra ODBC Connector is available for Microsoft® Windows®, Linux, and macOS platforms.

The *Installation and Configuration Guide* is suitable for users who are looking to access Cassandra data from their desktop environment. Application developers might also find the information helpful. Refer to your application for details on connecting via ODBC.

Note:

For information about how to use the connector in various BI tools, see the *Simba ODBC Connectors Quick Start Guide for Windows*: http://cdn.simba.com/docs/ODBC_QuickstartGuide/content/quick_start/intro.htm.

Windows Driver

Windows System Requirements

The Simba Cassandra ODBC Connector supports Apache Cassandra versions 3.x, 4.0, and 4.1.

Install the connector on client machines where the application is installed. Before installing the connector, make sure that you have the following:

Install the driver on client machines where the application is installed. Before installing the driver, make sure that you have the following:

- Administrator rights on your machine.
- A machine that meets the following system requirements:
  - One of the following operating systems:
    - Windows 10 or 8.1
    - Windows Server 2019, 2016, or 2012
  - 150 MB of available disk space

Before the connector can be used, the Visual C++ Redistributable for Visual Studio 2015 with the same bitness as the connector must also be installed. If you obtained the connector from the Simba website, then your installation of the connector automatically includes this dependency. Otherwise, you must install the redistributable manually. You can download the installation packages for the redistributable at https://www.microsoft.com/en-ca/download/details.aspx?id=48145.

Installing the Driver on Windows

If you did not obtain this driver from the Simba website, you might need to follow a different installation procedure. For more information, see the *Simba OEM ODBC Drivers Installation Guide*.

On 64-bit Windows operating systems, you can execute both 32-bit and 64-bit applications. However, 64-bit applications must use 64-bit drivers, and 32-bit applications must use 32-bit drivers. Make sure that you use a driver whose bitness matches the bitness of the client application:

- `Simba Cassandra 2.7 32-bit.msi` for 32-bit applications
- `Simba Cassandra 2.7 64-bit.msi` for 64-bit applications

You can install both versions of the on the same machine.

To install the Simba Cassandra ODBC Connector on Windows:

1. Depending on the bitness of your client application, double-click to run **Simba Cassandra 2.7 32-bit.msi** or **Simba Cassandra 2.7 64-bit.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**.
7. If you received a license file through email, then copy the license file into the `\lib` subfolder of the installation folder you selected above. You must have Administrator privileges when changing the contents of this folder.

Creating a Data Source Name on Windows

Typically, after installing the Simba Cassandra ODBC Connector, you need to create a Data Source Name (DSN).

Alternatively, for information about DSN-less connections, see Using a Connection String on page 44.

To create a Data Source Name on Windows:

1. From the Start menu, go to **ODBC Data Sources**.

   Note:

   Make sure to select the ODBC Data Source Administrator that has the same bitness as the client application that you are using to connect to Cassandra.

2. In the ODBC Data Source Administrator, click the **Drivers** tab, and then scroll down as needed to confirm that the Simba Cassandra ODBC Driver appears in the alphabetical list of ODBC drivers that are installed on your system.
3. Choose one:
   - To create a DSN that only the user currently logged into Windows can use, click the **User DSN** tab.
   - Or, to create a DSN that all users who log into Windows can use, click the **System DSN** tab.

> **Note:**
>
> It is recommended that you create a System DSN instead of a User DSN. Some applications load the data using a different user account, and might not be able to detect User DSNs that are created under another user account.

4. Click **Add**.
5. In the Create New Data Source dialog box, select **Simba Cassandra ODBC Driver** and then click **Finish**. The Simba Cassandra ODBC Driver DSN Setup dialog box opens.
6. In the **Data Source Name** field, type a name for your DSN.
7. Optionally, in the **Description** field, type relevant details about the DSN.
8. Choose one:
   - In the **Host** field, type the name or IP address of the host where your Cassandra instance is running.
   - Or, in the **Host** field, type a comma-separated list of host names or IP addresses of Cassandra servers to which the connector connects.

> **Note:**
>
> The connector attempts to connect to all the servers concurrently, and then keep the first connection that is successfully established. The connector does not maintain a connection with any of the other servers in the list.

9. In the **Port** field, type the number of the TCP port that the server uses to listen for client connections.

> **Note:**
>
> The default port used by Cassandra is 9042.

10. If user login is required to access the Cassandra instance, then configure authentication. For more information, see Configuring Authentication on Windows on page 11.
11. In the **Default Keyspace** field, type the name of the Cassandra keyspace to use by default.
12. To configure advanced connector options, click **Advanced Options**. For more information, see Configuring Advanced Options on Windows on page 12.

13. To configure logging behavior for the connector, click **Logging Options**. For more information, see Configuring Logging Options on Windows on page 17.
14. To test the connection, click **Test**. Review the results as needed, and then click **OK**.

> **Note:**
>
> If the connection fails, then confirm that the settings in the Simba Cassandra ODBC Driver DSN Setup dialog box are correct. Contact your Cassandra server administrator as needed.

15. To save your settings and close the Simba Cassandra ODBC Connector DSN Setup dialog box, click **OK**.
16. To close the ODBC Data Source Administrator, click **OK**.

Configuring Authentication on Windows

Some Cassandra databases require authentication for access, while all Astra databases require authentication. You can configure the Simba Cassandra ODBC Connector to authenticate your connection to the database by providing the necessary credentials. For more information, see the following:

- Configuring Authentication for Cassandra Databases on Windows on page 11
- Configuring Authentication for Astra Databases on Windows on page 12

Configuring Authentication for Cassandra Databases on Windows

Some Cassandra databases require you to authenticate your connection by providing a user name and password.

To configure authentication for Cassandra databases on Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, select the DSN, and then click **Configure**.
2. In the **Mechanism** drop-down list, select **User Name and Password**.
3. In the **Username** field, type the user name associated with the Cassandra database.
4. In the **Password** field, type the password corresponding to the user name you typed above.
5. To encrypt your credentials, select one of the following:
   - If the credentials are used only by the current Windows user, select **Current User Only**.

- Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.

6. To save your settings and close the dialog box, click **OK**.

Configuring Authentication for Astra Databases on Windows

To connect to an Astra database, you must authenticate your connection by providing a user name, password, and secure connection bundle.

To obtain the secure connection bundle associated with your Astra database, download it from the DataStax Constellation console. For more information, see "Obtaining database credentials" in the DataStax Astra documentation: https://docs.datastax.com/en/astra/aws/doc/dscloud/astra/dscloudObtainingCredentials.html.

To configure authentication for Astra databases on Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, select the DSN, and then click **Configure**.
2. In the **Mechanism** drop-down list, select **Cloud Secure Connection Bundle**.
3. In the **Username** field, type the user name associated with the database.
4. In the **Password** field, type the password corresponding to the user name you typed above.
5. In the **Secure Connection Bundle Path** field, type the full path and name of the secure connection bundle associated with your Astra database. To make sure that the DSN is compatible with all ODBC applications, escape the backslashes (`\`) in your file path by typing another backslash, for example, `C:\\temp`.
6. To encrypt your credentials, select one of the following:
   - If the credentials are used only by the current Windows user, select **Current User Only**.
   - Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.
7. To save your settings and close the dialog box, click **OK**.

Configuring Advanced Options on Windows

You can configure advanced options to modify the behavior of the connector.

To configure advanced options on Windows:

1. To access advanced options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Advanced Options**.

2. In the **Query Mode** list, select an option to specify how the connector 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**.
3. 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. For detailed information about each option, see the topic *Configuring data consistency* in the Apache Cassandra 3.0 documentation:https://docs.datastax.com/en/cassandra-oss/3.0/cassandra/dml/dmlDataConsistencyTOC.html.
4. In the **Load Balancing Policy** list, select the load balancing policy to use:
   - To cycle through all nodes in the cluster on a per-query basis, select **Round Robin**.
   - To first try all nodes in a primary "local" data center before trying any nodes from other data centers, select **DC Aware**.
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.

   Note:

   For more information about virtual tables, see Virtual Tables on page 49.

8. To use a Blacklist or Whitelist when connecting to hosts in the Cassandra cluster, enter the host IP addresses in the **Blacklist Hosts** or **Whitelist Hosts** field.
   - Each IP addresses should be entered in quotation marks, separated by a comma. For example: `"1.2.3.4","5.6.7.8"`.
9. To use a Blacklist or Whitelist of datacenter hosts, enter the host names or addresses in the **Blacklist Datacenter Hosts** or **Whitelist Datacenter Hosts** field.
   - Each name or addresses should be entered in quotation marks, separated by a comma. For example: `"datacenter1","datacenter2"`.

10. To use a token-aware policy to improve load balancing and latency, select the **Enable Token Aware** check box.
11. To use a latency-awareness algorithm to distribute more of the workload onto faster machines, select the **Enable Latency Aware** check box.
12. Select how the connector handles null value INSERT statements:
    - To configure the connector to insert all NULL values as specified in INSERT statements, select the **Enable null values insertion** check box.
    - To configure the connector to ignore NULL values inserted into a column that contains only NULL values, clear the **Enable null values insertion** check box.

> **Note:**
>
> For more information about this option, see Enable Null Value Insert on page 57.

13. Select how the connector handles capitalization in schema, table, and column names:
    - To differentiate between capital and lower-case letters in schema, table, and column names, select the **Enable Case Sensitive** check box.
    - To ignore the capitalization of schema, table, and column names, clear the **Enable Case Sensitive** check box.

> **Note:**
>
> For more information about case sensitivity in Cassandra, see Enable Case Sensitive on page 56.

14. To map text and varchar data types in Cassandra to use SQL_WVARCHAR, select the **Use SQL_WVARCHAR for string data type** check box.
15. Select how the connector handles large result sets:
    - To configure the connector 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 connector 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.

16. To configure client-server verification over SSL, use the options in the SSL area. For more information, see Configuring SSL Verification on Windows on page 15.
17. To save your settings and close the Advanced Options dialog box, click **OK**.
18. To close the Simba Cassandra ODBC Driver DSN Setup dialog box, click **OK**.

Configuring SSL Verification on Windows

If you are connecting to a Cassandra server that has Secure Sockets Layer (SSL) enabled, then you can configure the connector to connect to an SSL-enabled socket. When connecting to a server over SSL, the connector supports identity verification between the client and the server.

Note:

When connecting to Astra, two-way SSL verification is always enabled, and the required certificates are typically provided through the secure connection bundle.

Configuring an SSL Connection without Identity Verification

You can configure a connection that uses SSL but does not verify the identity of the client or the server.

To configure an SSL connection without verification on Windows:

1. To access the SSL options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Advanced Options**.
2. In the SSL area, select **One-way Server Verification** or **Two-way Server and Client Verification**.
3. Clear the **Enable Server Hostname Verification** check box.
4. To save your settings and close the dialog box, click **OK**.

Configuring One-way SSL Verification

You can configure one-way SSL verification so that the client verifies the identity of the Cassandra server.

To configure one-way SSL verification on Windows:

1. To access the SSL options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Advanced Options**.
2. In the SSL area, select **One-way Server Verification**.
3. Ensure that the **Enable Server Hostname Verification** check box is selected.
4. In the **Trusted CA Certificates** field, specify the full path of the PEM file containing the certificate for verifying the server.
5. To save your settings and close the dialog box, click **OK**.

Configuring Two-way SSL Verification

You can configure two-way SSL verification so that the client and the Cassandra server verify each other.

To configure two-way SSL verification on Windows:

1. To access the SSL options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Advanced Options**.
2. In the SSL area, select **Two-way Server and Client Verification**.
3. Ensure that the **Enable Server Hostname Verification** check box is selected.
4. In the **Trusted CA Certificates** field, specify the full path of the PEM file containing the certificate for verifying the server.
5. In the **Client-side Certificate** field, specify the full path of the PEM file containing the certificate for verifying the client.
6. In the **Client-side Private Key** field, specify the full path of the file containing the private key used to verify the client.
7. 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.

   > ⚠ **Important:**
   >
   > Passwords are saved in plain text in the DSN; they are not encrypted or censored.

8. To save your settings and close the dialog box, click **OK**.

Configuring Logging Options on Windows

To help troubleshoot issues, you can enable logging. In addition to functionality provided in the Simba Cassandra ODBC Connector, the ODBC Data Source Administrator provides tracing functionality.

Important:

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

Configuring Connector-wide Logging Options

The settings for logging apply to every connection that uses the Simba Cassandra ODBC Connector, so make sure to disable the feature after you are done using it.

To enable connector-wide logging on Windows:

1. To access logging options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Logging Options**.
2. From the **Log Level** drop-down list, select the logging level corresponding to the amount of information that you want to include in log files:

Logging Level	Description
OFF	Disables all logging.
FATAL	Logs severe error events that lead the connector to abort.
ERROR	Logs error events that might allow the connector to continue running.
WARNING	Logs events that might result in an error if action is not taken.
INFO	Logs general information that describes the progress of the connector.

Logging Level	Description
DEBUG	Logs detailed information that is useful for debugging the connector.
TRACE	Logs all connector activity.

3. In the **Log Path** field, specify the full path to the folder where you want to save log files.
4. If requested by 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. In the **Max Number Files** field, type the maximum number of log files to keep.

   Note:

   After the maximum number of log files is reached, each time an additional file is created, the connector deletes the oldest log file.

6. In the **Max File Size** field, type the maximum size of each log file in megabytes (MB).

   Note:

   After the maximum file size is reached, the connector creates a new file and continues logging.

7. Click **OK**.
8. Restart your ODBC application to make sure that the new settings take effect.

The Simba Cassandra ODBC Connector produces the following log files at the location you specify in the Log Path field, where *[DriverName]* is the name of the connector:

- A `Simba[DriverName]_driver.log` file that logs connector activity that is not specific to a connection.
- A `Simba[DriverName]_connection_[Number].log` for each connection made to the database, where *[Number]* is a number that identifies each log file. This file logs connector activity that is specific to the connection.

If you enable the `UseLogPrefix` connection property, the connector prefixes the log file name with the user name associated with the connection and the process ID of the

application through which the connection is made. For more information, see UseLogPrefix on page 79.

To disable connector logging on Windows:

1. Open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Logging Options**.
2. From the **Log Level** drop-down list, select **LOG_OFF**.
3. Click **OK**.
4. Restart your ODBC application to make sure that the new settings take effect.

Configuring FIPS on Windows

You can configure the FIPS (Federal Information Processing Standards) module to ensure the security, quality, and processing compatibility of various services.

To configure FIPS on Windows:

1. Locate `openssl.cnf` and the FIPS module binary in the following directory:

   ```
   <install_directory>/lib/OpenSSL<bitness>.DllA.
   ```

2. Execute the following command to generate the `fipsmodule.cnf` file:

   ```
   openssl fipsinstall -module <fips_dir>\fips.dll -out
   fipsmodule.cnf -provider_name fips.
   ```

3. In the `openssl.cnf` file, add:

   ```
   .include <path_to_fips_conf_file>/fipsmodule.cnf
   ```

   - Add a `[base_sect]` section with `activate = 1` under it.

   Note:

   Under `[provider_sect]`, make sure that there is `fips = fips_sect` and `base = base_sect`, which is used for base encoders and decoders.

4. Set the OPENSSL_CONF environment variable to the location of the `openssl.cnf` file.
5. Set the OPENSSL_MODULES environment variable to the directory containing the FIPS module binary.

Note:

- Install OpenSSL 3.0 on your machine.
- Make sure that all the fips module binary files are present in OPENSSL_MODULES path otherwise, the driver does not work as expected.

Verifying the Connector Version Number on Windows

If you need to verify the version of the Simba Cassandra ODBC Connector that is installed on your Windows machine, you can find the version number in the ODBC Data Source Administrator.

To verify the connector version number on Windows:

1. From the Start menu, go to **ODBC Data Sources**.

   Note:

   Make sure to select the ODBC Data Source Administrator that has the same bitness as the client application that you are using to connect to Cassandra.

2. Click the **Drivers** tab and then find the Simba Cassandra ODBC Connector in the list of ODBC Connectors that are installed on your system. The version number is displayed in the **Version** column.

macOS Driver

macOS System Requirements

The Simba Cassandra ODBC Connector supports Apache Cassandra versions 3.x, 4.0, and 4.1.

Install the connector on client machines where the application is installed. Each client machine that you install the connector on must meet the following minimum system requirements:

- One of the following macOS versions:
  - macOS 11 (Universal Binary – Intel and ARM support)
- 100MB of available disk space
- One of the following ODBC driver managers installed:
  - iODBC 3.52.9 or later
  - unixODBC 2.2.14 or later

Installing the Driver on macOS

If you did not obtain this connector from the Simba website, you might need to follow a different installation procedure. For more information, see the *Simba OEM ODBC Drivers Installation Guide*.

The Simba Cassandra ODBC Connector is available for macOS as a `.dmg` file named `Simba Cassandra 2.7.dmg`. The connector supports both 32- and 64-bit client applications.

To install the Simba Cassandra ODBC Connector on macOS:

1. Double-click **Simba Cassandra 2.7.dmg** to mount the disk image.
2. In the installer, click **Continue**.
3. On the Software License Agreement screen, click **Continue**, and when the prompt appears, click **Agree** if you agree to the terms of the License Agreement.
4. Optionally, to change the installation location, click **Change Install Location**, then select the desired location, and then click **Continue**.

> **Note:**
>
> By default, the connectors files are installed in the `/Library/simba/cassandraodbc` directory.

5. To accept the installation location and begin the installation, click **Install**.
6. When the installation completes, click **Close**.
7. If you received a license file through email, then copy the license file into the `/lib` subfolder in the connectors installation directory. You must have root privileges when changing the contents of this folder.

   For example, if you installed the connectors to the default location, you would copy the license file into the `/Library/simba/cassandraodbc/lib` folder.

Next, configure the environment variables on your machine to make sure that the ODBC driver manager can work with the connectors . For more information, see Configuring the ODBC Driver Manager on Non-Windows Machines on page 29.

Configuring FIPS on macOS

You can configure the FIPS (Federal Information Processing Standards) module to ensure the security, quality, and processing compatibility of various services.

To configure FIPS on macOS:

1. Locate the `openssl.cnf` and the FIPS module binary in the following directory:

   ```
   <install_directory>/lib.
   ```

2. Execute the following command to generate the `fipsmodule.cnf` file:

   ```
   openssl fipsinstall -module <fips_dir>\fips.dylib -out
   fipsmodule.cnf -provider_name fips.
   ```

3. In the `openssl.cnf` file, add:

   ```
   .include <path_to_fips_conf_file>/fipsmodule.cnf
   ```

   - Add a `[base_sect]` section with `activate = 1` under it.

   Note:

   Under `[provider_sect]`, make sure that there is `fips = fips_sect` and `base = base_sect`, which is used for base encoders and decoders.

4. Set the OPENSSL_CONF environment variable to the location of the `openssl.cnf` file.
5. Set the OPENSSL_MODULES environment variable to the location of the FIPS module binary.

Note:

- Install OpenSSL 3.0 on your machine.
- Make sure that all the fips module binary files are present in OPENSSL_MODULES path otherwise, the driver does not work as expected.

Verifying the Driver Version Number on macOS

If you need to verify the version of the Simba Cassandra ODBC Connector that is installed on your macOS machine, you can query the version number through the Terminal.

To verify the driver version number on macOS:

- At the Terminal, run the following command:

```
pkgutil --info com.simba.cassandraodbc
```

The command returns information about the Simba Cassandra ODBC Connector that is installed on your machine, including the version number.

Linux Driver

The Linux connector is available as an RPM file and as a tarball package. If you are installing the connector on a Debian machine, you must use the Debian package.

Linux System Requirements

The Simba Cassandra ODBC Connector supports Apache Cassandra versions 3.x, 4.0, and 4.1.

Install the on client machines where the application is installed. Each client machine that you install the on must meet the following minimum system requirements:

- One of the following distributions:
  - Red Hat® Enterprise Linux® (RHEL) 7 or 8
  - CentOS 7 or 8
  - SUSE Linux Enterprise Server (SLES) 12 or 15
  - Debian 8 or 9
  - Ubuntu 16.04, 18.04, or 20.04
- 50MB of available disk space
- One of the following ODBC driver managers installed:
  - iODBC 3.52.9 or later
  - unixODBC 2.2.14 or later

To install the , you must have root access on the machine.

Installing the Driver Using the RPM File

If you did not obtain this connector from the Simba website, you might need to follow a different installation procedure. For more information, see the *Simba OEM ODBC Installation Guide*.

On 64-bit editions of Linux, you can execute both 32- and 64-bit applications. However, 64-bit applications must use 64-bit connectors, and 32-bit applications must use 32-bit connectors. Make sure that you use a connector whose bitness matches the bitness of the client application:

- `simbacassandra-[Version]-[Release].i686.rpm` for the 32-bit connector
- `simbacassandra-[Version]-[Release].x86_64.rpm` for the 64-bit connector

The placeholders in the file names are defined as follows:

- *[Version]* is the version number of the connector
- *[Release]* is the release number for this version of the

You can install both the 32-bit and 64-bit versions of the connector on the same machine.

To install the Simba Cassandra ODBC Connector using the RPM File:

1. Log in as the root user.
2. Navigate to the folder containing the RPM package for the .
3. Depending on the Linux distribution that you are using, run one of the following commands from the command line, where *[RPMFileName]* is the file name of the RPM package:
   - If you are using Red Hat Enterprise Linux or CentOS, run the following command:

     ```
     yum --nogpgcheck localinstall [RPMFileName]
     ```

   - Or, if you are using SUSE Linux Enterprise Server, run the following command:

     ```
     zypper install [RPMFileName]
     ```

   The Simba Cassandra ODBC Connector files are installed in the `/opt/simba/cassandraodbc` directory.
4. If you received a license file through email, then copy the license file into the `/opt/simba/cassandraodbc/lib/32` or `/opt/simba/cassandraodbc/lib/64` folder, depending on the version of the connector that you installed.

Next, configure the environment variables on your machine to make sure that the ODBC driver manager can work with the . For more information, see Configuring the ODBC Driver Manager on Non-Windows Machines on page 29.

Installing the Connector Using the Tarball Package

If you did not obtain this connector from the Simba website, you might need to follow a different installation procedure. For more information, see the *Simba OEM ODBC Connectors Installation Guide*.

The Simba Cassandra ODBC Connector is available as a tarball package named `SimbaCassandraODBC-[Version].[Release]-Linux.tar.gz`, where *[Version]* is the version number of the connector and *[Release]* is the release number for this version of the connector. The package contains both the 32-bit and 64-bit versions of the connector.

On 64-bit editions of Linux, you can execute both 32- and 64-bit applications. However, 64-bit applications must use 64-bit connectors, and 32-bit applications must use 32-bit connectors. Make sure that you use a connector whose bitness matches the bitness of the client application. You can install both versions of the connector on the same machine.

To install the connector using the tarball package:

1. Log in as the root user, and then navigate to the folder containing the tarball package.
2. Run the following command to extract the package and install the connector:

   ```
   tar -zxvf [TarballName]
   ```

   Where *[TarballName]* is the name of the tarball package containing the connector.

   The Simba Cassandra ODBC Connector files are installed in the `/opt/simba/cassandraodbc` directory.
3. If you received a license file through email, then copy the license file into the `opt/simba/cassandraodbc/lib/32` or `opt/simba/cassandraodbc/lib/64` folder, depending on the version of the connector that you installed.

Next, configure the environment variables on your machine to make sure that the ODBC driver manager can work with the connector. For more information, see Configuring the ODBC Driver Manager on Non-Windows Machines on page 29.

Installing the Driver on Debian

To install the driver on a Debian machine, use the Debian package instead of the RPM file or tarball package.

On 64-bit editions of Debian, you can execute both 32- and 64-bit applications. However, 64-bit applications must use 64-bit drivers, and 32-bit applications must use 32-bit drivers. Make sure that you use the version of the driver that matches the bitness of the client application:

- `simba_[Version]-[Release]_i386.deb` for the 32-bit connector
- `simba_[Version]-[Release]_amd64.deb` for the 64-bit connector

[Version] is the version number of the connector, and *[Release]* is the release number for this version of the connector.

You can install both versions of the connector on the same machine.

To install the Simba Cassandra ODBC Connector on Debian:

1. Log in as the root user, and then navigate to the folder containing the Debian package for the connector.
2. Double-click **cassandra_*[Version]-[Release]*_i386.deb** or **cassandra_*[Version]-[Release]*_amd64.deb**.
3. Follow the instructions in the installer to complete the installation process.

   The Simba Cassandra ODBC Connector files are installed in the `/opt/simba/cassandraodbc` directory.
4. If you received a license file via email, then copy the license file into the `/opt/simba/cassandraodbc/lib/32` or `/opt/simba/cassandraodbc/lib/64` folder, depending on the version of the connector that you installed. You must have root privileges when changing the contents of this folder.

Next, configure the environment variables on your machine to make sure that the ODBC driver manager can work with the connector. For more information, see Configuring the ODBC Driver Manager on Non-Windows Machines on page 29.

Configuring FIPS on Linux

You can configure the FIPS (Federal Information Processing Standards) module to ensure the security, quality, and processing compatibility of various services.

To configure FIPS on Linux:

1. Locate `openssl.cnf` and the FIPS module binary in the following directory:

   ```
   <install_directory>/lib/.
   ```
2. Execute the following command to generate the `fipsmodule.cnf` file:

   ```
   openssl fipsinstall -module <fips_dir>/fips.so -out
   fipsmodule.cnf -provider_name fips.
   ```
3. In the `openssl.cnf` file, add:

   ```
   .include <path_to_fips_conf_file>/fipsmodule.cnf
   ```

   - Add a `[base_sect]` section with `activate = 1` under it.

Note:

Under `[provider_sect]`, make sure that there is `fips = fips_sect` and `base = base_sect`, which is used for base encoders and decoders.

4. Set the OPENSSL_CONF environment variable to the location of the `openssl.cnf` file.
5. Set the OPENSSL_MODULES environment variable to the directory containing the FIPS module binary.

Note:

- Install OpenSSL 3.0 on your machine.
- Make sure that all the fips module binary files are present in OPENSSL_MODULES path otherwise, the driver does not work as expected.

Verifying the Driver Version Number on Linux

If you need to verify the version of the Simba Cassandra ODBC Connector that is installed on your Linux machine, you can query the version number through the command-line interface if the connector was installed using an RPM file.

To verify the driver version number on Linux using the command-line interface:

- Depending on your package manager, at the command prompt, run one of the following commands:
  - `yum list | grep CassandraODBC`
  - `rpm -qa | grep CassandraODBC`

The command returns information about the Simba Cassandra ODBC Connector that is installed on your machine, including the version number.

Configuring the ODBC Driver Manager on Non-Windows Machines

To make sure that the ODBC driver manager on your machine is configured to work with the Simba Cassandra ODBC Connector, do the following:

- Set the library path environment variable to make sure that your machine uses the correct ODBC driver manager. For more information, see Specifying ODBC Driver Managers on Non-Windows Machines on page 29.
- If the connector configuration files are not stored in the default locations expected by the ODBC driver manager, then set environment variables to make sure that the driver manager locates and uses those files. For more information, see Specifying the Locations of the Driver Configuration Files on page 30.

After configuring the ODBC driver manager, you can configure a connection and access your data store through the connector.

Specifying ODBC Driver Managers on Non-Windows Machines

You need to make sure that your machine uses the correct ODBC driver manager to load the connector. To do this, set the library path environment variable.

macOS

If you are using a macOS machine, then set the DYLD_LIBRARY_PATH environment variable to include the paths to the ODBC driver manager libraries. For example, if the libraries are installed in `/usr/local/lib`, then run the following command to set DYLD_LIBRARY_PATH for the current user session:

```
export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:/usr/local/lib
```

For information about setting an environment variable permanently, refer to the macOS shell documentation.

Linux

If you are using a Linux machine, then set the LD_LIBRARY_PATH environment variable to include the paths to the ODBC driver manager libraries. For example, if the libraries are installed in `/usr/local/lib`, then run the following command to set LD_LIBRARY_PATH for the current user session:

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

For information about setting an environment variable permanently, refer to the Linux shell documentation.

Specifying the Locations of the Driver Configuration Files

By default, ODBC driver managers are configured to use hidden versions of the `odbc.ini` and `odbcinst.ini` configuration files (named `.odbc.ini` and `.odbcinst.ini`) located in the home directory, as well as the `simba.cassandraodbc.ini` file in the `lib` subfolder of the connector installation directory. If you store these configuration files elsewhere, then you must set the environment variables described below so that the driver manager can locate the files.

If you are using iODBC, do the following:

- Set ODBCINI to the full path and file name of the `odbc.ini` file.
- Set ODBCINSTINI to the full path and file name of the `odbcinst.ini` file.
- Set CASSANDRAODBC to the full path and file name of the `simba.cassandraodbc.ini` file.

If you are using unixODBC, do the following:

- Set ODBCINI to the full path and file name of the `odbc.ini` file.
- Set ODBCSYSINI to the full path of the directory that contains the `odbcinst.ini` file.
- Set CASSANDRAODBC to the full path and file name of the `simba.cassandraodbc.ini` file.

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

For iODBC:

```
export ODBCINI=/usr/local/odbc/odbc.ini
export ODBCINSTINI=/usr/local/odbc/odbcinst.ini
export CASSANDRAODBC=/etc/simba.cassandraodbc.ini
```

For unixODBC:

```
export ODBCINI=/usr/local/odbc/odbc.ini
export ODBCSYSINI=/usr/local/odbc
export CASSANDRAODBC=/etc/simba.cassandraodbc.ini
```

To locate the `simba.cassandraodbc.ini` file, the driver uses the following search order:

1. If the CASSANDRAODBC environment variable is defined, then the connector searches for the file specified by the environment variable.
2. The connector searches the directory that contains the connector library files for a file named `simba.cassandraodbc.ini`.
3. The connector searches the current working directory of the application for a file named `simba.cassandraodbc.ini`.
4. The connector searches the home directory for a hidden file named `.simba.cassandraodbc.ini` (prefixed with a period).
5. The connector searches the `/etc` directory for a file named `simba.cassandraodbc.ini`.

Configuring ODBC Connections on a Non-Windows Machine

The following sections describe how to configure ODBC connections when using the Simba Cassandra ODBC Connector on non-Windows platforms:

Creating a Data Source Name on a Non-Windows Machine

When connecting to your data store using a DSN, you only need to configure the `odbc.ini` file. Set the properties in the `odbc.ini` file to create a DSN that specifies the connection information for your data store. For information about configuring a DSN-less connection instead, see Configuring a DSN-less Connection on a Non-Windows Machine on page 34.

If your machine is already configured to use an existing `odbc.ini` file, then update that file by adding the settings described below. Otherwise, copy the `odbc.ini` file from the `Setup` subfolder in the connector installation directory to the home directory, and then update the file as described below.

To create a Data Source Name on a non-Windows machine:

1. In a text editor, open the `odbc.ini` configuration file.

   Note:

   If you are using a hidden copy of the `odbc.ini` file, you can remove the period (`.`) from the start of the file name to make the file visible while you are editing it.

2. Create a section that has the same name as your DSN, and then specify configuration options as key-value pairs in the section:
   a. Set the `Driver` property to the full path of the connector library file that matches the bitness of the application.

      For example, on a macOS machine:

```
Driver=/Library/
simba/cassandraodbc/lib/libcassandraodbc_sbu.dylib
```

As another example, for a 32-bit connector on a Linux machine:

```
Driver=/opt/
simba/cassandraodbc/lib/32/libcassandraodbc_sb32.so
```

b. Do one of the following:

- If you are connecting to a single Cassandra server, set the `Host` property to the IP address or host name of the server, and then set the `Port` property to the number of the TCP port that the server uses to listen for client connections.

  For example:

  ```
  Host=192.168.222.160
  Port=9042
  ```

- Or, if you are connecting to a multiple servers, set the `Host` property to a comma-separated list of the servers in the cluster, specifying the host names or IP addresses and port numbers.

  For example:

  ```
  Host=192.168.222.160:9042, 192.168.222.165:9042,
  192.168.222.231:9042
  ```

c. If authentication is required to access the server, then enable authentication and specify your credentials. For more information, see Configuring Authentication on a Non-Windows Machine on page 37.

d. If you want to connect to the server through SSL, then enable SSL and specify the certificate information. For more information, see Configuring SSL Verification on a Non-Windows Machine on page 38.

e. Optionally, set additional key-value pairs as needed to specify other optional connection settings. For detailed information about all the configuration options supported by the Simba Cassandra ODBC Connector, see Connector Configuration Properties on page 54.

3. Save the `odbc.ini` configuration file.

Note:

If you are storing this file in its default location in the home directory, then prefix the file name with a period (`.`) so that the file becomes hidden. If you are storing this file in another location, then save it as a non-hidden file (without the prefix), and make sure that the ODBCINI environment variable specifies the location. For more information, see Specifying the Locations of the Driver Configuration Files on page 30.

For example, the following is an `odbc.ini` configuration file for macOS containing a DSN that connects to a single Cassandra server without authentication:

```
[ODBC Data Sources]
Sample DSN=Simba Cassandra ODBC Connector
[Sample DSN]
Driver=/Library/simba/cassandraodbc/lib/libcassandraodbc_
sbu.dylib
Host=192.168.222.160
Port=9042
```

As another example, the following is an `odbc.ini` configuration file for a 32-bit connector on a Linux machine, containing a DSN that connects to a single Cassandra server with authentication:

```
[ODBC Data Sources]
Sample DSN=Simba Cassandra ODBC Connector 32-bit
[Sample DSN]
Driver=/opt/simba/cassandraodbc/lib/32/libcassandraodbc_
sb32.so
Host=192.168.222.160
Port=9042
```

You can now use the DSN in an application to connect to the data store.

Configuring a DSN-less Connection on a Non-Windows Machine

To connect to your data store through a DSN-less connection, you need to define the connector in the `odbcinst.ini` file and then provide a DSN-less connection string in your application.

If your machine is already configured to use an existing `odbcinst.ini` file, then update that file by adding the settings described below. Otherwise, copy the

`odbcinst.ini` file from the `Setup` subfolder in the connector installation directory to the home directory, and then update the file as described below.

To define a driver on a non-Windows machine:

1. In a text editor, open the `odbcinst.ini` configuration file.

   > **Note:**
   >
   > If you are using a hidden copy of the `odbcinst.ini` file, you can remove the period (`.`) from the start of the file name to make the file visible while you are editing it.

2. In the `[ODBC Drivers]` section, add a new entry by typing a name for the connector, an equal sign (=), and then `Installed`.

   For example:

   ```
   [ODBC Drivers]
   =Installed
   ```

3. Create a section that has the same name as the connector (as specified in the previous step), and then specify the following configuration options as key-value pairs in the section:
   a. Set the `Driver` property to the full path of the connector library file that matches the bitness of the application.

      For example, on a macOS machine:

      ```
      Driver=/Library/
      simba/cassandraodbc/lib/libcassandraodbc_sbu.dylib
      ```

      As another example, for a 32-bit connector on a Linux machine:

      ```
      Driver=/opt/
      simba/cassandraodbc/lib/32/libcassandraodbc_sb32.so
      ```

   b. Optionally, set the `Description` property to a description of the connector.

      For example:

      ```
      Description=Simba Cassandra ODBC Connector
      ```

4. Save the `odbcinst.ini` configuration file.

Note:

If you are storing this file in its default location in the home directory, then prefix the file name with a period (`.`) so that the file becomes hidden. If you are storing this file in another location, then save it as a non-hidden file (without the prefix), and make sure that the ODBCINSTINI or ODBCSYSINI environment variable specifies the location. For more information, see Specifying the Locations of the Driver Configuration Files on page 30.

For example, the following is an `odbcinst.ini` configuration file for macOS:

```
[ODBC Drivers]
Description= Simba Cassandra ODBC Connector
Driver=/Library/simba/cassandraodbc/lib/libcassandraodbc_
sbu.dylib
```

As another example, the following is an `odbcinst.ini` configuration file for both the 32- and 64-bit connectors on Linux:

```
[ODBC Drivers]
Driver=/opt/simba/cassandraodbc/lib/32/libcassandraodbc_
sb32.so
Description=Simba Cassandra ODBC Connector(64 bit)
Driver=/opt/simba/cassandraodbc/lib/64/libcassandraodbc_
sb64.so
```

You can now connect to your data store by providing your application with a connection string where the `Driver` property is set to the connector name specified in the `odbcinst.ini` file, and all the other necessary connection properties are also set. For more information, see "DSN-less Connection String Examples" in Using a Connection String on page 44.

For instructions about configuring specific connection features, see the following:

- Configuring Authentication on a Non-Windows Machine on page 37
- Configuring SSL Verification on a Non-Windows Machine on page 38

For detailed information about all the connection properties that the connector supports, see Connector Configuration Properties on page 54.

Configuring Authentication on a Non-Windows Machine

Some Cassandra databases require authentication for access, while all Astra databases require authentication. You can configure the Simba Cassandra ODBC Connector to authenticate your connection to the database by providing the necessary credentials. For more information, see the following:

- Configuring Authentication for Cassandra Databases on a Non-Windows Machine on page 37
- Configuring Authentication for Astra Databases on a Non-Windows Machine on page 37

You can set the connection properties described below in a connection string or in a DSN (in the `odbc.ini` file). Settings in the connection string take precedence over settings in the DSN.

Configuring Authentication for Cassandra Databases on a Non-Windows Machine

Some Cassandra databases require you to authenticate your connection by providing a user name and password.

To configure authentication for Cassandra databases:

1. Set the `AuthMech` property to `1`.
2. Set the `UID` property to the user name associated with the Cassandra server.
3. Set the `PWD` property to password corresponding to the user name you provided above.

Configuring Authentication for Astra Databases on a Non-Windows Machine

To connect to an Astra database, you must authenticate your connection by providing a user name, password, and secure connection bundle.

To obtain the secure connection bundle associated with your Astra database, download it from the DataStax Constellation console. For more information, see "Obtaining database credentials" in the DataStax Astra documentation: https://docs.datastax.com/en/astra/aws/doc/dscloud/astra/dscloudObtainingCredentials.html.

To configure authentication for Astra databases:

1. Set the `AuthMech` property to `2`.
2. Set the `UID` property to the user name associated with the database.
3. Set the `PWD` property to password corresponding to the user name you provided above.

4. Set the `SecureConnectionBundlePath` property to the full path and name of the secure connection bundle associated with your Astra database. To make sure that the DSN or connection string is compatible with all ODBC applications, escape the backslashes (`\`) in your file path by typing another backslash, for example, `C:\\temp`.

Configuring SSL Verification on a Non-Windows Machine

If you are connecting to a Cassandra server that has Secure Sockets Layer (SSL) enabled, then you can configure the connector to connect to an SSL-enabled socket. When connecting to a server over SSL, the connector supports identity verification between the client and the server.

> **Note:**
>
> When connecting to Astra, two-way SSL verification is always enabled, and the required certificates are typically provided through the secure connection bundle.

You can set the connection properties described below in a connection string or in a DSN (in the `odbc.ini` file). Settings in the connection string take precedence over settings in the DSN.

Configuring a Connection without SSL

You can configure a connection that does not use SSL.

To configure a connection without SSL on a non-Windows machine:

- Set the `SSLMode` property to `0`.

Configuring One-way SSL Verification

You can enable the client to verify the Cassandra server.

To configure one-way SSL verification on a non-Windows machine:

1. Set the `SSLMode` property to `1`.
2. Set the `UseSslIdentityCheck` property to `1`.
3. Set the `SSLTrustedCertsPath` property to the full path of the `.pem` file containing the certificate for verifying the server.

Configuring Two-way SSL Verification

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

To configure two-way SSL verification on a non-Windows machine:

1. Set the `SSLMode` property to `2`.
2. Set the `UseSslIdentityCheck` property to `1`.
3. Set the `SSLTrustedCertsPath` property to the full path of the `.pem` file containing the certificate for verifying the server.
4. Set the `SSLUserCertsPath` property to the full path of the `.pem` file containing the certificate for verifying the client.
5. Set the `SSLUserKeyPath` property to the full path of the file containing the private key used to verify the client.
6. If the private key file is protected with a password, set the `SSLUserKeyPWD` property to specify the password.

> **Important:**
>
> Passwords are saved in plain text in the DSN; they are not encrypted or censored.

Configuring an SSL Connection that does not Verify Certificates

You can configure a connection that uses SSL but does not verify the client or the server.

To configure an SSL connection without verification on a non-Windows machine:

1. Set the `SSLMode` property to `1` or `2`.
2. Set the `UseSslIdentityCheck` property to `0`.

Configuring Logging Options on a Non-Windows Machine

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

> **Important:**
>
> - Only enable logging long enough to capture an issue. Logging decreases performance and can consume a large quantity of disk space.
> - The settings for logging apply to every connection that uses the Simba Cassandra ODBC Connector, so make sure to disable the feature after you are done using it.

Logging is configured through connector-wide settings in the `simba.cassandraodbc.ini` file, which apply to all connections that use the connector.

To enable logging on a non-Windows machine:

1. Open the `simba.cassandraodbc.ini` configuration file in a text editor.
2. To specify the level of information to include in log files, set the `LogLevel` property to one of the following numbers:

LogLevel Value	Description
0	Disables all logging.
1	Logs severe error events that lead the connector to abort.
2	Logs error events that might allow the connector to continue running.
3	Logs events that might result in an error if action is not taken.
4	Logs general information that describes the progress of the connector.
5	Logs detailed information that is useful for debugging the connector.
6	Logs all connector activity.

3. Set the `LogPath` key to the full path to the folder where you want to save log files.
4. Set the `LogFileCount` key to the maximum number of log files to keep.

> **Note:**
>
> After the maximum number of log files is reached, each time an additional file is created, the connector deletes the oldest log file.

5. Set the `LogFileSize` key to the maximum size of each log file in bytes.

> **Note:**
>
> After the maximum file size is reached, the connector creates a new file and continues logging.

6. Optionally, to prefix the log file name with the user name and process ID associated with the connection, set the `UseLogPrefix` property to `1`.
7. Save the `simba.cassandraodbc.ini` configuration file.
8. Restart your ODBC application to make sure that the new settings take effect.

The Simba Cassandra ODBC Connector produces the following log files at the location you specify using the `LogPath` key, where *[ConnectorName]* is the name of the connector:

- A `Simba`*`[ConnectorName]`*`_connector.log` file that logs connector activity that is not specific to a connection.
- A `Simba`*`[connectorName]`*`_connection_`*`[Number]`*`.log` for each connection made to the database, where *[Number]* is a number that identifies each log file. This file logs connector activity that is specific to the connection.

If you set the `UseLogPrefix` property to `1`, then each file name is prefixed with *`[UserName]_[ProcessID]_`*, where *[UserName]* is the user name associated with the connection and *[ProcessID]* is the process ID of the application through which the connection is made. For more information, see UseLogPrefix on page 79.

To disable logging on a non-Windows machine:

1. Open the `simba.cassandraodbc.ini` configuration file in a text editor.
2. Set the `LogLevel` key to `0`.
3. Save the `simba.cassandraodbc.ini` configuration file.
4. Restart your ODBC application to make sure that the new settings take effect.

Testing the Connection on a Non-Windows Machine

To test the connection, you can use an ODBC-enabled client application. For a basic connection test, you can also use the test utilities that are packaged with your driver manager installation. For example, the iODBC driver manager includes simple utilities called iodbctest and iodbctestw. Similarly, the unixODBC driver manager includes simple utilities called isql and iusql.

Using the iODBC Driver Manager

You can use the iodbctest and iodbctestw utilities to establish a test connection with your connector. Use iodbctest to test how your connector works with an ANSI application, or use iodbctestw to test how your connector works with a Unicode application.

Note:

There are 32-bit and 64-bit installations of the iODBC driver manager available. If you have only one or the other installed, then the appropriate version of iodbctest (or iodbctestw) is available. However, if you have both 32- and 64-bit versions installed, then you need to make sure that you are running the version from the correct installation directory.

For more information about using the iODBC driver manager, see http://www.iodbc.org.

To test your connection using the iODBC driver manager:

1. Run **iodbctest** or **iodbctestw**.
2. Optionally, if you do not remember the DSN, then type a question mark (`?`) to see a list of available DSNs.
3. Type the connection string for connecting to your data store, and then press ENTER. For more information, see Using a Connection String on page 44.

If the connection is successful, then the `SQL>` prompt appears.

Using the unixODBC Driver Manager

You can use the isql and iusql utilities to establish a test connection with your connector and your DSN. isql and iusql can only be used to test connections that use a DSN. Use isql to test how your connector works with an ANSI application, or use iusql to test how your connector works with a Unicode application.

Note:

There are 32-bit and 64-bit installations of the unixODBC driver manager available. If you have only one or the other installed, then the appropriate version of isql (or iusql) is available. However, if you have both 32- and 64-bit versions installed, then you need to make sure that you are running the version from the correct installation directory.

For more information about using the unixODBC driver manager, see http://www.unixodbc.org.

To test your connection using the unixODBC driver manager:

- Run isql or iusql by using the corresponding syntax:

- `isql [DataSourceName]`
- `iusql [DataSourceName]`

`[DataSourceName]` is the DSN that you are using for the connection.

If the connection is successful, then the `SQL>` prompt appears.

Note:

For information about the available options, run isql or iusql without providing a DSN.

Using a Connection String

For some applications, you might need to use a connection string to connect to your data source. For detailed information about how to use a connection string in an ODBC application, refer to the documentation for the application that you are using.

The connection strings in the following sections are examples showing the minimum set of connection attributes that you must specify to successfully connect to the data source. Depending on the configuration of the data source and the type of connection you are working with, you might need to specify additional connection attributes. For detailed information about all the attributes that you can use in the connection string, see Connector Configuration Properties on page 54.

DSN Connection String Example

The following is an example of a connection string for a connection that uses a DSN:

```
DSN=[DataSourceName]
```

[DataSourceName] is the DSN that you are using for the connection.

You can set additional configuration options by appending key-value pairs to the connection string. Configuration options that are passed in using a connection string take precedence over configuration options that are set in the DSN.

DSN-less Connection String Examples

Some applications provide support for connecting to a data source using a connector without a DSN. To connect to a data source without using a DSN, use a connection string instead.

The placeholders in the examples are defined as follows, in alphabetical order:

- *[BundlePath]* is the full path and name of the secure connection bundle associated with your Astra database
- *[PortNumber]* is the number of the TCP port that the server uses to listen for client connections.
- *[Server]* is the IP address or host name of the Cassandra or Astra server to which you are connecting. You can specify a comma-separated list of servers.
- *[YourPassword]* is the password corresponding to your user name.
- *[YourUserName]* is the user name that you use to access the server.

Connecting to a Cassandra Server Without Authentication

The following is the format of a DSN-less connection string for a Cassandra server that does not require authentication:

```
Driver=Simba Cassandra ODBC Driver;Host=[Server];Port=
[PortNumber];
```

For example:

```
Driver=Simba Cassandra ODBC
Driver;Host=192.168.222.160;Port=9042;
```

Connecting to a Cassandra Server Requiring Authentication

The following is the format of a DSN-less connection string for a Cassandra server that requires authentication:

```
Driver=Simba Cassandra ODBC Driver;Host=[Server];Port=
[PortNumber];AuthMech=1;UID=[YourUserName];
PWD=[YourPassword];
```

For example:

```
Driver=Simba Cassandra ODBC
Driver;Host=192.168.222.160;Port=9042;AuthMech=1;UID=skroob;P
WD=12345;
```

Connecting to an Astra Server

The following is the format of a DSN-less connection string for an Astra server:

```
Driver=Simba Cassandra ODBC Driver;AuthMech=2;UID=
[YourUserName];PWD=[YourPassword];SecureConnectionBundlePath=
[BundlePath];
```

For example:

```
Driver=Simba Cassandra ODBC
Driver;AuthMech=2;UID=skroob;PWD=12345;SecureConnectionBundle
Path="C:\\Users\\admin\\Documents\\Astra\\secure-connect-
simba_database.zip";
```

Features

For more information on the features of the Simba Cassandra ODBC Connector, see the following:

SQL Connector

The SQL Connector feature of the connector allows applications to execute standard SQL queries against Cassandra. It converts SQL-92 queries to CQL operations and processes the results. When the connector is configured to work in SQL with CQL Fallback mode, it uses the SQL Connector to handle SQL queries by loading and processing the data in memory. This feature enables the connector to support SQL operations that cannot be executed natively through CQL queries, such as column filtering and table joins.

Data Types

The Simba Cassandra ODBC Connector can convert between Cassandra data types and SQL data types.

The table below lists the supported ODBC 3.x data type mappings. A few data types are mapped to a different type when using ODBC 2.x. Those data type mappings are listed in the next table.

To support complex data types such as sets, lists, and maps, the connector renormalizes the data into virtual tables. For more information, see Virtual Tables on page 49.

Cassandra Type	SQL Type
ASCII	SQL_VARCHAR

Cassandra Type	SQL Type
BIGINT	SQL_INT
BLOB	SQL_LONGVARBINARY
BOOLEAN	SQL_BIT
COUNTER	SQL_BIGINT
DATE	SQL_DATE (2.x) and SQL_TYPE_DATE (3.x)
DECIMAL	SQL_DECIMAL
DOUBLE	SQL_DOUBLE
FLOAT	SQL_REAL
INET	SQL_VARCHAR
INT	SQL_INTEGER
SMALLINT	SQL_SMALLINT
TEXT	SQL_WVARCHAR
TIME	SQL_TIME (2.x) or SQL_TYPE_TIME (3.x)
TIMESTAMP See the note below.	SQL_TYPE_TIMESTAMP
TIMEUUID	GUID
TINYINT	SQL_TINYINT
UUID	GUID
VARCHAR	SQL_VARCHAR
VARINT	SQL_NUMERIC

Note:

Cassandra internally represents a 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 Simba Cassandra ODBC Connector is from "1601-01-01 00:00:00.000" to "9999-12-31 23:59:59.999".

Cassandra Type	SQL Type
TIMESTAMP See the note above.	SQL_TIMESTAMP
TIMEUUID	SQL_VARCHAR
UUID	SQL_VARCHAR

Note:

The Simba Cassandra ODBC Connector no longer supports Apache Cassandra version 2.x.

User-Defined Types

The Simba Cassandra ODBC Connector provides support for Cassandra user-defined types in certain situations. Specifically, user-defined types are supported as base data or as part of a collection.

The column names for user-defined types are constructed in the following format:

[Column_name]_[Subtype_name]

Where:

- *[Column_name]* is the name of the column that contains the user-defined type
- *[Subtype_name]*is the name of the subtype in the user-defined type

For example, a user-defined type that contains a user's full name is created as follows:

```
CREATE TYPE fullname
(first_name text, last_name text)
```

The connector creates a table that contains two columns, `id:text` and `name:fullname`. The connector then reads the data from this table as follows:

```
"id", "name_first_name", "name_last_name"
"a", "Chris", "Kwan"
```

Virtual Tables

One advantage of the Apache Cassandra design is the ability to store data that is denormalized into fewer tables. By taking advantage of nested data structures such as sets, lists, and maps, transactions can be simplified. However, the ODBC interface does not natively support accessing this type of data. By renormalizing the data contained within collections (sets, lists, and maps) into virtual tables, the Simba Cassandra ODBC 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 connector creates the following virtual tables:

- A "base" table, which contains the same data as the real table except for the collection columns.
- A virtual table for each collection column, which expands the nested data.

Virtual tables refer to the data in the real table, enabling the connector 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.

The base table and virtual tables appear as additional tables in the list of tables that exist in the database. The base 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_** by default), and the name of the column.

For example, consider the table below. ExampleTable is a Cassandra database table that contains an integer primary key column named pk_int, a list column, a map column, and a set column (named StringSet).

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 connector would generate multiple virtual tables to represent this single table. The first virtual table is the base table, shown below.

pk_int
1
3

The base table contains the same data as the original database table except for the collections, which are omitted from this table and expanded in other virtual tables.

The following tables show the virtual tables that renormalize the data from the List, Map, and StringSet columns.

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

pk_int	Map#key	Map#value
1	S1	A
1	S2	b
3	S1	t

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 or map. 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 connector handles 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, use the following query:

```
INSERT INTO "ExampleTable_vt_List" (pk_int, "List#index",
"List#value") VALUES (3, -1, '106')
```

Write-Back

The Simba Cassandra ODBC Connector supports Data Manipulation Languages (DML) statements such as INSERT, UPDATE, and DELETE.

Because Cassandra supports the UPSERT operation instead of INSERT and UPDATE, when you execute an INSERT or UPDATE statement using the Simba Cassandra ODBC Connector, the resulting behavior is an UPSERT operation. When you use the connector to write data to a Cassandra database, the INSERT and UPDATE operations both set the column value regardless of whether the data already exists.

You can use the TRUNCATE TABLE statement to delete rows from non-virtual tables. However, to delete rows from virtual tables, you must use the DELETE FROM statement instead.

Query Modes

The Simba Cassandra ODBC Connector can be configured to process queries as SQL statements or as CQL statements.

The default query mode used by the connector is SQL with CQL Fallback. In this query mode, the connector treats all incoming queries as SQL. If an error occurs while handling the query as SQL, then the connector passes the original query to Cassandra to execute as CQL. However, because Cassandra is not aware of virtual tables, incoming queries that reference virtual tables fail when they are passed through to the server to be executed as CQL.

Alternatively, you can configure the connector to work in SQL mode or CQL mode. When working in SQL mode, the connector treats all incoming queries as SQL, so any queries that are not written in standard SQL-92 syntax will fail. When working in CQL mode, the connector treats all incoming queries as CQL, so any queries written in a non-CQL syntax will fail.

Catalog and Schema Support

The Simba Cassandra ODBC Connector supports both catalogs and schemas to make it easy for the connector to work with various ODBC applications. Since Cassandra only organizes column families into keyspaces, the connector provides a synthetic catalog named CASSANDRA under which all of the keyspaces are organized. The connector also maps the ODBC schema to the Cassandra keyspace.

Security and Authentication

To protect data from unauthorized access, some Cassandra data stores require connections to be authenticated with user credentials or encrypted using the SSL protocol. The Simba Cassandra ODBC Connector provides full support for these authentication protocols.

Note:

In this documentation, "SSL" refers to both TLS (Transport Layer Security) and SSL (Secure Sockets Layer). The connector supports TLS 1.1 and 1.2. The SSL version used for the connection is the highest version that is supported by both the connector and the server.

The connector provides a mechanism that enables you to authenticate your connection using your Cassandra user name and password. The connector also supports authentication to Astra databases, which require a user name, password, and secure connection bundle. For detailed configuration instructions, see Configuring

Authentication on Windows on page 11 or Configuring Authentication on a Non-Windows Machine on page 37.

Additionally, the connector supports the following types of SSL connections:

- No identity verification
- One-way authentication
- Two-way authentication

Depending on the configuration of your Cassandra server, you might have the option of connecting without using SSL encryption. However, Astra instances always require SSL encryption with two-way authentication.

It is recommended that you enable SSL whenever you connect to a server that is configured to support it. SSL encryption protects data and credentials when they are transferred over the network, and provides stronger security than authentication alone. For detailed configuration instructions, see Configuring SSL Verification on Windows on page 15 or Configuring SSL Verification on a Non-Windows Machine on page 38.

Connector Configuration Properties

Connector Configuration Options lists the configuration options available in the Simba Cassandra ODBC Connector alphabetically by field or button label. Options having only key names, that is, not appearing in the user interface of the connector, are listed alphabetically by key name.

When creating or configuring a connection from a Windows machine, the fields and buttons described below are available in the following dialog boxes:

- Simba Cassandra ODBC Driver DSN Setup
- Advanced Options
- Logging Options

When using a connection string or configuring a connection from a non-Windows machine, use the key names provided below.

Configuration Options Appearing in the User Interface

The following configuration options are accessible via the Windows user interface for the Simba Cassandra ODBC Connector, or via the key name when using a connection string or configuring a connection from a Linux or macOS computer:

Binary Column Length

Key Name	Default Value	Required
`BinaryColumnLength`	`4000`	No

Description

The default column length to report for BLOB columns.

Blacklist Datacenter Hosts

Key Name	Default Value	Required
`BlacklistDatacenterFilteringHosts`	None	No

Description

The address or name of data center hosts in the Cassandra cluster you do not wish to connect to. Each name or addresses should be entered in quotation marks, separated by a comma.

For example: `"datacenter1","datacenter2"`.

Blacklist Hosts

Key Name	Default Value	Required
`BlacklistFilteringHosts`	None	No

Description

The IP addresses of data store hosts in the Cassandra cluster you do not wish to connect to. Each IP addresses should be entered in quotation marks, separated by a

comma.

For example: `"1.2.3.4","5.6.7.8"`.

Client-side Certificate

Key Name	Default Value	Required
`SSLUserCertsPath`	None	Yes, if two-way SSL verification is enabled.

Description

The full path to the `.pem` file containing the certificate for verifying the client.

Client-side Private Key

Key Name	Default Value	Required
`SSLUserKeyPath`	None	Yes, if two-way SSL verification is enabled.

Description

The full path to the file containing the private key used to verify the client.

Default Keyspace

Key Name	Default Value	Required
`DefaultKeyspace`	None	No

Description

The default keyspace (schema) to connect to in Cassandra.

Enable Case Sensitive

Key Name	Default Value	Required
`EnableCaseSensitive`	Selected (`1`)	No

Description

This option specifies whether the connector differentiates between capital and lower-case letters in schema, table, and column names.

- Enabled (`1`): The connector differentiates between capital and lower-case letters in schema, table, and column names. It is recommended that you enclose the names of all schemas, tables, and columns in double quotation marks (") if this option is enabled.
- Disabled (`0`): The connector ignores the capitalization of schema, table, and column names.

⚠ Important:

- If the naming conventions for your Cassandra server are case-sensitive, you must leave this option enabled.
- If you are using the connector in a BI tool such as Tableau or Lumira, it is recommended that you leave this option enabled.
- If this option is disabled, then queries that use case-sensitive schema, table, and column names are not supported.

Enable Latency Aware

Key Name	Default Value	Required
`EnableLatencyAware`	Clear (`0`)	No

Description

This option specifies whether the connector uses a latency-awareness algorithm to distribute the load away from slower-performing nodes.

- Enabled (`1`): The connector uses the latency-awareness algorithm.
- Disabled (`0`): The connector does use the latency-awareness algorithm.

Enable Null Value Insert

Key Name	Default Value	Required
`EnableNullInsert`	Clear (`0`)	No

Description

This option specifies how the connector inserts NULL values.

- Enabled (`1`): The connector inserts all NULL values as specified in INSERT statements.
- Disabled (`0`): If an INSERT statement only specifies NULL values for a column or does not specify any values for a column, then the connector omits that column when executing the INSERT statement.

Consider the following before modifying this property:

- It is recommended that you leave the property disabled so that the connector does not insert NULL values into empty cells and create tombstones, which may decrease server performance and cause errors to occur. However, this setting may decrease connector performance when executing INSERT statements that affect a large number of rows.
- It is recommended that you enable this property by setting it to `1` only when executing INSERT statements that do not contain unnecessary NULL values, because inserting NULL values into empty columns creates tombstones.

For more information about tombstones, see *About deletes* in the Apache Cassandra 3.0 documentation: https://docs.datastax.com/en/cassandra-oss/3.0/cassandra/dml/dmlAboutDeletes.html?hl=deletes.

Enable Paging

Key Name	Default Value	Required
`EnablePaging`	Selected (`1`)	No

Description

This option specifies whether to split large result sets into pages.

- Enabled (`1`): The connector splits large result sets into pages.
- Disabled (`0`): The connector fetches all results into memory regardless of the result set size.

See also the connector configuration option Rows Per Page on page 66.

Enable Server Hostname Verification

Key Name	Default Value	Required
`UseSslIdentityCheck`	Selected (`1`)	No

Description

This option specifies whether the connector requires the host name of the server to match the host name in the SSL certificate.

- Enabled (`1`): During SSL verification the connector requires the host name of the server to match the host name in the certificate.
- Disabled (`0`): During SSL verification the connector allows the host name of the server to not match the host name in the certificate.

Enable Token Aware

Key Name	Default Value	Required
`EnableTokenAware`	Selected (`1`)	No

Description

This option specifies whether to use a token-aware policy to improve load balancing and latency.

- Enabled (`1`): The connector uses the token-aware policy.
- Disabled (`0`): The token-aware policy is not used.

Encrypt Password

Key Name	Default Value	Required
N/A	All Users Of This Machine	No

Description

This option specifies how the connector encrypts the credentials that are saved in the DSN:

- **Current User Only**: The credentials are encrypted, and can only be used by the current Windows user.
- **All Users Of This Machine**: The credentials are encrypted, but can be used by any user on the current Windows machine.

Important:

This option is available only when you configure a DSN using the Simba ODBC Driver DSN Setup dialog box in the Windows connector. When you connect to the data store using a connection string, the connector does not encrypt your credentials.

Host

Key Name	Default Value	Required
`Host`	None	

Description

The IP address or host name of the Cassandra server.

You can specify a comma-separated list of IP addresses or host names. The connector attempts to connect to each of the servers in succession, and establishes the first available connection.

Key File Password

Key Name	Default Value	Required
`SSLUserKeyPWD`	None	No

Description

The password for the private key file that is specified in the Client-side Private Key field or the `SSLUserKeyPath` key.

For more information, see Client-side Private Key on page 56.

Load Balancing Policy

Key Name	Default Value	Required
`LoadBalancingPolicy`	DC Aware (`0`)	No

Description

This option specifies the load balancing policy to be used.

- Round Robin (`1`): The connector uses the Round Robin policy to cycle through all nodes in the cluster on a per-query basis.
- DC Aware (`0`): The connector uses the DC Aware policy. For each query, all nodes in a primary "local" data center are tried first, before any nodes from other data centers.

Note:

As of connector version 2.5.6, the `COLoadBalancingPolicy` key has been deprecated and replaced by the `LoadBalancingPolicy` key. The connector still accepts `COLoadBalancingPolicy` in the connection string, but this key may not be supported in future releases. It is recommended that you use the `LoadBalancingPolicy` key instead.

Log Level

Key Name	Default Value	Required
`LogLevel`	OFF (`0`)	No

Description

Use this property to enable or disable logging in the connector and to specify the amount of detail included in log files.

Important:

- Only enable logging long enough to capture an issue. Logging decreases performance and can consume a large quantity of disk space.
- The settings for logging apply to every connection that uses the Simba Cassandra ODBC Connector, so make sure to disable the feature after you are done using it.
- This option is not supported in connection strings. To configure logging for the Windows connector, you must use the Logging Options dialog box. To configure logging for a non-Windows connector, you must use the `simba.cassandraodbc.ini` file.

Set the property to one of the following values:

- OFF (`0`): Disable all logging.
- FATAL (`1`): Logs severe error events that lead the connector to abort.
- ERROR (`2`): Logs error events that might allow the connector to continue running.
- WARNING (`3`): Logs events that might result in an error if action is not taken.
- INFO (`4`): Logs general information that describes the progress of the connector.
- DEBUG (`5`): Logs detailed information that is useful for debugging the connector.
- TRACE (`6`): Logs all connector activity.

When logging is enabled, the connector produces the following log files at the location you specify in the Log Path (`LogPath`) property, where *[ConnectorName]* is the name of the connector:

- A `Simba`*`[ConnectorName]`*`_driver.log` file that logs connector activity that is not specific to a connection.
- A `Simba`*`[ConnectorName]`*`_connection_`*`[Number]`*`.log` for each connection made to the database, where *[Number]* is a number that identifies each log file. This file logs connector activity that is specific to the connection.

If you enable the `UseLogPrefix` connection property, the connector prefixes the log file name with the user name associated with the connection and the process ID of the application through which the connection is made. For more information, see UseLogPrefix on page 79.

Log Path

Key Name	Default Value	Required
`LogPath`	None	Yes, if logging is enabled.

Description

The full path to the folder where the connector saves log files when logging is enabled.

Important:

This option is not supported in connection strings. To configure logging for the Windows connector, you must use the Logging Options dialog box. To configure logging for a non-Windows connector, you must use the `simba.cassandraodbc.ini` file.

Max File Size

Key Name	Default Value	Required
`LogFileSize`	`20971520`	No

Description

The maximum size of each log file in bytes. After the maximum file size is reached, the connector creates a new file and continues logging.

If this property is set using the Windows UI, the entered value is converted from megabytes (MB) to bytes before being set.

Important:

This option is not supported in connection strings. To configure logging for the Windows connector, you must use the Logging Options dialog box. To configure logging for a non-Windows connector, you must use the `simba.cassandraodbc.ini` file.

Max Number Files

Key Name	Default Value	Required
LogFileCount	50	No

Description

The maximum number of log files to keep. After the maximum number of log files is reached, each time an additional file is created, the connector deletes the oldest log file.

Important:

This option is not supported in connection strings. To configure logging for the Windows connector, you must use the Logging Options dialog box. To configure logging for a non-Windows connector, you must use the `simba.cassandraodbc.ini` file.

Mechanism

Key Name	Default Value	Required
AuthMech	No Authentication (0)	No

Description

The authentication mechanism to use.

Select one of the following settings, or set the key to the corresponding number:

- No Authentication (0)
- User Name And Password (1)
- Cloud Secure Connection Bundle (2)

Password

Key Name	Default Value	Required
PWD	None	Yes, if the authentication mechanism is User Name And Password (1) or Cloud Secure Connection Bundle (2).

Description

The password corresponding to the user name that you provided in the User Name field (the UID key).

Port

Key Name	Default Value	Required
Port	9042	Yes, unless connecting to an Astra cloud database.

Description

The number of the TCP port that the Cassandra server uses to listen for client connections.

Query Mode

Key Name	Default Value	Required
QueryMode	SQL with CQL Fallback (2)	No

Description

This option specifies the query mode to use when sending queries to Cassandra.

- SQL (0):The connector uses SQL_QUERY_MODE and executes all queries in SQL.
- CQL (1): The connector uses CQL_QUERY_MODE and executes all queries in CQL.

- SQL with CQL Fallback (`2`): The connector uses SQL_WITH_CQL_FALLBACK_QUERY_MODE and executes all queries in SQL by default. If a query fails, then the connector executes the query in CQL.

Rows Per Page

Key Name	Default Value	Required
`RowsPerPage`	`10000`	No

Description

When the Enable Paging option is enabled, use this option to specify the maximum number of rows to display on each page.

See also the connector configuration option Enable Paging on page 58.

Secure Connection Bundle Path

Key Name	Default Value	Required
`SecureConnectionBundlePath`	None	Yes, if connecting to an Astra database and `SecureConnectionBundleUrl` is not set.

Description

The full path and name of the secure connection bundle associated with your Astra database.

Note:

- If both `SecureConnectionBundlePath` and `SecureConnectionBundleUrl` are specified, `SecureConnectionBundlePath` takes precedence.
- To make sure that the connection string is compatible with all ODBC applications, escape the backslashes (`\`) in your file path by typing another backslash, for example, `C:\\temp`.

You can download a secure connection bundle from the DataStax Constellation console. For more information, see "Obtaining database credentials" in the DataStax Astra documentation:

https://docs.datastax.com/en/astra/aws/doc/dscloud/astra/dscloudObtainingCredentials.html.

SSL

Key Name	Default Value	Required
`SSLMode`	No SSL (`0`)	No

Description

This option specifies how the connector uses SSL to connect to the Cassandra server.

- No SSL (`0`): The connector does not use SSL.
- One-way Server Verification (`1`): If the Enable Server Hostname Verification option is enabled, the client verifies the Cassandra server using SSL. Otherwise, the connector connects to the Cassandra server using SSL but the client and the server do not verify each other.
- Two-way Server and Client Verification (`2`): If the Enable Server Hostname Verification option is enabled, the client and the Cassandra server verify each other using SSL. Otherwise, the connector connects to the Cassandra server using SSL but the client and the server do not verify each other.

For more information, see Enable Server Hostname Verification on page 59.

String Column Length

Key Name	Default Value	Required
`StringColumnLength`	`4000`	No

Description

The default column length to report for ASCII, TEXT, and VARCHAR columns.

Trusted CA Certificates

Key Name	Default Value	Required
SSLTrustedCertsPath	The path to the cacerts.pem file in the \lib folder in the connector's installation directory. The exact file path varies depending on the version of the connector that is installed.	No

Description

The full path to the .pem file containing the certificate for verifying the server.

Tunable Consistency

Key Name	Default Value	Required
TunableConsistency	ONE (1)	No

Description

The specific Cassandra replica or the number of Cassandra replicas that must process a query in order for the query to be considered successful.

Select one of the following settings, or set the key to the number corresponding to the desired setting:

- ANY (0)
- ONE (1)
- TWO (2)
- THREE (3)
- QUORUM (4)
- ALL (5)
- LOCAL_QUORUM (6)

- EACH_QUORUM (7)
- LOCAL_ONE (10)

These settings correspond to the consistency levels available in Cassandra. For detailed information about each consistency level, see *Configuring data consistency* in the Apache Cassandra 3.0 documentation: https://docs.datastax.com/en/cassandra-oss/3.0/cassandra/dml/dmlDataConsistencyTOC.html.

Use SQL_WVARCHAR For String Data Types

Key Name	Default Value	Required
UseSqlWVarchar	Clear (0)	No

Description

This option specifies how text and varchar types are mapped to SQL.

- Enabled (1): The Cassandra text and varchar types are mapped to SQL_WVARCHAR.
- Disabled (0): The Cassandra text and varchar types are mapped to SQL_VARCHAR.

User Name

Key Name	Default Value	Required
UID	None	Yes, if the authentication mechanism is User Name And Password (1) or Cloud Secure Connection Bundle (2).

Description

The user name that you use to access the Cassandra server.

Virtual Table Name Separator

Key Name	Default Value	Required
VTTableNameSeparator	_vt_	No

Description

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`

Whitelist Datacenter Hosts

Key Name	Default Value	Required
`WhitelistDatacenterFilteringHosts`	None	No

Description

The addresses or names of the datacenter hosts in the Cassandra cluster you wish to connect to. Each name or addresses should be entered in quotation marks, separated by a comma.

For example: `"datacenter1","datacenter2"`.

Whitelist Hosts

Key Name	Default Value	Required
`WhitelistFilteringHosts`	None	No

Description

The IP addresses of data store hosts in the Cassandra cluster you wish to connect to. Each IP addresses should be entered in quotation marks, separated by a comma.

For example: `"1.2.3.4","5.6.7.8"`.

Configuration Options Having Only Key Names

The following configuration options do not appear in the Windows user interface for the Simba Cassandra ODBC Connector. They are accessible only when you use a connection string or configure a connection on macOS or Linux.

- Cached Rows Limit on page 71
- Concurrent Requests on page 72

The `UseLogPrefix` property must be configured as a Windows Registry key value, or as a connector-wide property in the `simba.cassandraodbc.ini` file for macOS or Linux.

Cached Rows Limit

Key Name	Default Value	Required
`CachedRowsLimit`	`5000`	No

Description

The maximum number of rows that the connector caches before forcing a flush.

Concurrent Requests

Key Name	Default Value	Required
`NumConcurrentRequests`	`100`	No

Description

The number of concurrent requests per insertion thread.

Core Connections Per Host

Key Name	Default Value	Required
`CoreConnectionsPerHost`	`1`	No

Description

The number of connections that the connector makes to each server in each IO thread.

Default Column Scale

Key Name	Default Value	Required
`DefaultColumnScale`	`10`	No

Description

The default scale used for decimal columns.

Disable Decimal Padding

Key Name	Default Value	Required
`DisableDecimalPadding`	Clear (`0`)	No

Description

This option specifies whether to disable decimal padding.

- Enabled (`1`): The connector disables decimal padding, and fits the decimal scale to the input parameter's scale, on a per-row basis.

- Trailing zeroes are discarded. For example, 1.500 is inserted as 1.5.
- Values with a scale that is greater than the default column scale are truncated. For example, if the default column scale is 3 and the value is 1.5557, the value is inserted as 1.555.

- Disabled (`0`): The connector uses decimal padding.

To set the default decimal column scale, see Default Column Scale on page 72.

Driver

Key Name	Default Value	Required
`Driver`	`Cassandra ODBC Driver` when installed on Windows, or the absolute path of the connector shared object file when installed on a non-Windows machine.	Yes

Description

On Windows, the name of the installed connector (`Cassandra ODBC Driver;`).

On other platforms, the name of the installed connector as specified in `odbcinst.ini`, or the absolute path of the connector shared object file.

Enable Asynchronous Writes

Key Name	Default Value	Required
`EnableAsynchronousWrites`	Clear (`0`)	No

Description

This option specifies whether to enable asynchronous database write.

- Enabled (`1`): The connector allows asynchronous database writes.
- Disabled (`0`): The connector does not allow asynchronous writes.

Enable Compress Swap File

Key Name	Default Value	Required
EnableCompressSwapFile	0	No

Description

This option specifies whether the SQLEngine setting compresses data internally, per connection.

We recommend that this option be enabled if the data can be compressed, and if the queries that are run often cause the SQLEngine to reach its memory limit and swap to disk.

- Enabled (1): The connector compresses TemporaryTable data internally.
- Disabled (0): The connector does not compress TemporaryTable data internally.

Enable Joins

Key Name	Default Value	Required
EnableJoins	Selected (1)	No

Description

This option specifies whether SQL joins are allowed.

- Enabled (1): The connector allows joins, and executes SQL queries that contain them.
- Disabled (0): The connector does not allow joins. If a SQL query that contains a join is submitted, the connector returns an error.

ForceInPredPassdown

Key Name	Default Value	Required
ForceInPredPassdown	0	No

Description

This option specifies whether the connector passes down the filtering of the IN predicate to the server. For more info on this clause, see the Cassandra

documentation: https://cassandra.apache.org/doc/latest/cassandra/cql/dml.html#select.

- Enabled (`1`): The connector forces the passdown of the IN predicate to the server.
- Disabled (`0`): If the datasource applies the filter, the connector passes down filters to the server.

Insert Query Threads

Key Name	Default Value	Required
`NumInsertQueryThreads`	`2`	No

Description

The number of insert query threads.

Iterations Per Insert Thread

Key Name	Default Value	Required
`NumIterationsPerInsertThread`	`50`	No

Description

The number of iterations for each insert query thread.

IO Threads

Key Name	Default Value	Required
`NumThreadsIO`	`1`	No

Description

The number of IO threads, that is, the number of threads that handle query requests.

Maximum Concurrent Connections

Key Name	Default Value	Required
`MaxConcurrentCreation`	`1`	No

Description

The maximum number of connections that can exist concurrently.

A new connection is created when the existing connections are unable to keep up with request throughput.

Maximum Concurrent Requests

Key Name	Default Value	Required
`MaxConcurrentRequestsThreshold`	`100`	No

Description

The maximum number of concurrent requests that can exist on a connection before the connector creates a new connection.

> **Note:**
>
> If the number of connections has reached the Maximum Connections Per Host value, a new connection is not created.

Maximum Connections Per Host

Key Name	Default Value	Required
`MaxConnectionsPerHost`	`2`	No

Description

The maximum number of connections that the connector makes to each server in each IO thread.

A new connection is created when the existing connections are unable to keep up with request throughput.

Maximum Requests Per Flush

Key Name	Default Value	Required
`MaxRequestsPerFlush`	`128`	No

Description

The maximum number of requests processed by an IO worker per flush.

Pending Requests High Water Mark

Key Name	Default Value	Required
PendingRequestsHighWaterMark	256	No

Description

The high water mark for the number of requests that can be queued for a connection in a connection pool.

If the number of queued requests exceeds this value, the connector disables writes to a host on an IO worker until the number of queued requests drops below the low water mark.

Pending Requests Low Water Mark

Key Name	Default Value	Required
PendingRequestsLowWaterMark	128	No

Description

The low water mark for the number of requests queued for a connection in a connection pool.

If the number of queued requests exceeds the high water mark value (see Pending Requests High Water Mark on page 77), the connector disables writes to a host on an IO worker until the number of queued requests drops below this value.

Queue Size Event

Key Name	Default Value	Required
QueueSizeEvent	8192	No

Description

The size of the fixed-size queue that stores events.

Queue Size IO

Key Name	Default Value	Required
`QueueSizeIO`	`8192`	No

Description

The size of the fixed-size queue that stores pending requests.

Return UDT As String

Key Name	Default Value	Required
`ReturnUdtAsString`	`0`	No

Description

This option specifies how the connector handles user-defined types (UDT).

- Enabled (`1`): The connector returns UDT columns as a VARCHAR type.
- Disabled (`0`): The connector exposes UDT sub-types as individual columns of the table.

Important:

When this option is enabled, the connector cannot perform INSERT operations and returns an error.

Secure Connection Bundle Url

Key Name	Default Value	Required
`SecureConnectionBundleUrl`	None	No

Description

The full URL to the secure connection bundle associated with your Astra database.

Note:

- If both `SecureConnectionBundlePath` and `SecureConnectionBundleUrl` are specified, `SecureConnectionBundlePath` takes precedence.
- To make sure that the connection string is compatible with all ODBC applications, escape any slashes (`/`) or backslashes (`\`) in your file path by typing another backslash.

You can download a secure connection bundle from the DataStax Constellation console. For more information, see "Obtaining database credentials" in the DataStax Astra documentation: https://docs.datastax.com/en/astra/aws/doc/dscloud/astra/dscloudObtainingCredentials.html.

UseLogPrefix

Key Name	Default Value	Required
`UseLogPrefix`	`0`	No

Description

This option specifies whether the connector includes a prefix in the names of log files so that the files can be distinguished by user and application.

Set the property to one of the following values:

- `1`: The connector prefixes log file names with the user name and process ID associated with the connection that is being logged.

  For example, if you are connecting as a user named "jdoe" and using the connector in an application with process ID 7836, the generated log files would be named `jdoe_7836_Simba`*`[ConnectorName]`*`_connector.log` and `jdoe_7836_Simba`*`[ConnectorName]`*`_connection_`*`[Number]`*`.log`, where *[Number]* is a number that identifies each connection-specific log file.
- `0`: The connector does not include the prefix in log file names.

To configure this option for the Windows connector, you create a value for it in one of the following registry keys:

- For a 32-bit connector installed on a 64-bit machine: **HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Simba\Simba Cassandra ODBC**

Connector\Driver

- Otherwise: **HKEY_LOCAL_MACHINE\SOFTWARE\Simba\Simba Cassandra ODBC Connector\Driver**
- For a 32-bit connector installed on a 64-bit machine: **HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Simba\\Driver**
- Otherwise: **HKEY_LOCAL_MACHINE\SOFTWARE\\Driver**

Use `UseLogPrefix` as the value name, and either `0` or `1` as the value data.

To configure this option for a non-Windows connector, you must use the `simba.cassandraodbc.ini` file.

Write Bytes High Water Mark

Key Name	Default Value	Required
`WriteBytesHighWaterMark`	`65536`	No

Description

The high water mark for the number of bytes that can be outstanding on a connection.

If the number of bytes outstanding on a connection exceeds this value, the connector disables writes to a host on an IO worker until the number of outstanding bytes drops below the low water mark.

Write Bytes Low Water Mark

Key Name	Default Value	Required
`WriteBytesLowWaterMark`	`32768`	No

Description

The low water mark for the number of bytes that can be outstanding on a connection.

If the number of bytes outstanding on a connection exceeds the high water mark (see Write Bytes High Water Mark on page 80), the connector disables writes to a host on an IO worker until the number of outstanding bytes drops below this value.

Third-Party Trademarks

Linux is the registered trademark of Linus Torvalds in Canada, United States and/or other countries.

Mac, macOS, Mac OS, and OS X are trademarks or registered trademarks of Apple, Inc. or its subsidiaries in Canada, United States and/or other countries.

Microsoft, MSDN, Windows, Windows Server, Windows Vista, and the Windows start button are trademarks or registered trademarks of Microsoft Corporation or its subsidiaries in Canada, United States and/or other countries.

Red Hat, Red Hat Enterprise Linux, and CentOS are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in Canada, United States and/or other countries.

SUSE is a trademark or registered trademark of SUSE LLC or its subsidiaries in Canada, United States and/or other countries.

Apache Cassandra, Apache, and Cassandra are trademarks of The Apache Software Foundation or its subsidiaries in Canada, the United States and/or other countries.

All other trademarks are trademarks of their respective owners.