Magnitude Simba MySQL ODBC Data Connector

Installation and Configuration Guide

Version 1.1.8

July 2023

Copyright

Contact Us

Magnitude Software, Inc.

www.magnitude.com

About This Guide

Purpose

The *Magnitude Simba MySQL ODBC Data Connector Installation and Configuration Guide* explains how to install and configure the Magnitude Simba MySQL 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 MySQL ODBC Connector, as well as administrators and developers integrating the connector.

Knowledge Prerequisites

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

- Familiarity with the platform on which you are using the Simba MySQL ODBC Connector
- Ability to use the data source to which the Simba MySQL 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 MySQL ODBC Connector

The Simba MySQL ODBC Connector enables Business Intelligence (BI), analytics, and reporting on data that is stored in MySQL 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 *Installation and Configuration Guide* is suitable for users who are looking to access MySQL 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 MySQL ODBC Connector supports MySQL versions 5.7 and 8.0.

Install the connector on client machines where the application is installed. Before installing the connector, 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
    - 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 MySQL 1.1 32-bit.msi` for 32-bit applications
- `Simba MySQL 1.1 64-bit.msi` for 64-bit applications

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

To install the Simba MySQL ODBC Connector on Windows:

1. Depending on the bitness of your client application, double-click to run **Simba MySQL 1.1 32-bit.msi** or **Simba MySQL 1.1 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 MySQL 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 MySQL.

2. In the ODBC Data Source Administrator, click the **Drivers** tab, and then scroll down as needed to confirm that the Simba MySQL ODBC Connector appears in the alphabetical list of ODBC connectors 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 MySQL ODBC Connector** and then click **Finish**. The Simba MySQL ODBC Connector 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. In the **User** field, type an appropriate user name for accessing the MySQL database.
9. If you are required to authenticate the connection using the caching_sha2_ password, or SHA-256 authentication plugin, then in the **Password** field, type the password corresponding to the user name you typed above.
10. Optionally, if you are authenticating the connection using the caching_sha2_ password or SHA-256 authentication plugin, specify an RSA public key for encrypting your password by doing the following:
    a. Select the **SSL** tab.
    b. In the **RSA Public Key** field, specify the full path and name of the file containing the key.

> **Note:**
>
> If you do not specify an RSA public key or configure SSL for your connection, the connector automatically retrieves a public key from the server and uses it to encrypt your password.

11. In the **Host** field, type the name or IP address of the MySQL server.
12. 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 MySQL is 3306.

13. In the **Catalog** field, type the name of the database that you want to connect to by default.

    Note:

    You can still issue queries on other databases by specifying the database schema in your query statement.

14. To configure logging behavior for the connector, click **Logging Options**. For more information, see Configuring Logging Options on Windows on page 14.
15. To configure additional connector options, select a tab:
    - Select the **Connection** tab to configure how the connector handles connections and specify connector behavior that applies at connection time. For more information, see Configuring Connection Options on Windows on page 11.
    - Select the **Metadata** tab to configure how the connector handles specific types of data. For more information, see Configuring Metadata Options on Windows on page 11.
    - Select the **Cursor/Results** tab to configure how the connector handles cursors and query results. For more information, see Configuring Cursor and Result Options on Windows on page 12.
    - Select the **SSL** tab to configure encryption and identity verification using SSL. For more information, see Configuring SSL Connections on Windows on page 17.
    - Select the **Misc** tab to configure other additional connector options. For more information, see Configuring Additional Connector Options on Windows on page 13.
16. 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 MySQL ODBC Driver DSN Setup dialog box are correct. Contact your MySQL server administrator as needed.

17. To save your settings and close the Simba MySQL ODBC Driver DSN Setup dialog box, click **OK**.
18. To close the ODBC Data Source Administrator, click **OK**.

Configuring Connection Options on Windows

You can configure options to modify how the connector handles connections and specify connector behavior that applies at connection time.

To configure connection options on Windows:

1. To access the connection options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then select the **Connection** tab.
2. To compress the communication between the client and the server, select the **Use Compression** check box.

   > **Note:**
   >
   > Compression may reduce the amount of network traffic for certain workflows, but in some cases it may increase CPU usage.

3. To automatically reconnect to the server when a communication link error occurs, select the **Enable Automatic Reconnect** check box.
4. To prevent the connector from opening dialog boxes to prompt for additional information during connection time, select the **Don't Prompt When Connecting** check box.
5. To enable support for batched statements, select the **Allow Multiple Statements** check box.
6. To allow sandboxed connections through which you can reset the password after attempting to connect using an expired password, select the **Can Handle Expired Password** check box.
7. To close the connection after it has been inactive for the amount of time specified by the `interactive_timeout` setting on the server, select the **Interactive Client** check box.
8. In the **Initial Statement** field, type a statement for the connector to execute immediately after connecting to the server.
9. To save your settings and close the Simba MySQL ODBC Driver DSN Setup dialog box, click **OK**.

Configuring Metadata Options on Windows

You can configure how the connector handles specific types of data.

To configure metadata options on Windows:

1. To access the metadata options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then select the **Metadata** tab.
2. To return BIGINT data as SQL_INTEGER data instead of SQL_BIGINT data, select the **Treat BIGINT Columns As INT Columns** check box.
3. To return Binary columns as Character columns, select the **Always Handle Binary Function Results As Character Data** check box.
4. To allow column names in queries to include the database name (for example, using the format *[Database].[Table].[Column]*), select the **Ignore Schema In Column Specifications** check box.

   Note:

   If this option is disabled, the connector returns an error if a column name specified in a statement includes the database name.

5. To return fully-qualified column names that include the table name when the `SQLDescribeCol()` function is called, select the **Include Table Name In SQLDescribeCol()** check box.
6. To limit column sizes to 32-bit signed values, select the **Limit Column Size To A Signed 32-bit Value** check box.
7. Optionally, to configure the connector to recognize table type information from the data source, select the **Enable Table Types** check box. For more information, see Enable Table Types on page 56.
8. Optionally, to return default metadata as SQL_VARCHAR, select the **Return Default metadata for SQLDescribeParam** check box.
9. To save your settings and close the Simba MySQL ODBC Driver DSN Setup dialog box, click **OK**.

Configuring Cursor and Result Options on Windows

You can configure how the connector handles cursors and query results.

To configure cursor and result options on Windows:

1. To access the cursor and result options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then select the **Cursor/Results** tab.

2. To reduce memory consumption by preventing the connector from caching result sets into memory, select the **Don't Cache Results Of Forward-Only Cursors** check box.
3. To specify what the connector returns after DML operations, do one of the following:
   - To return the number of matched rows, select the **Return Matched Rows Instead Of Affected Rows** check box.
   - Or, to return the number of affected rows, clear the **Return Matched Rows Instead Of Affected Rows** check box.
4. To respect the `sql_auto_is_null` setting specified in the server instead of automatically disabling it, select the **Enable SQL_AUTO_IS_NULL** check box.

   Note:

   For more information about `sql_auto_is_null`, see "Server System Variables" in the *MySQL Reference Manual*: https://dev.mysql.com/doc/refman/5.7/en/server-systemvariables.html#sysvar_sql_auto_is_null.

5. To specify how the connector handles date values where the month or day is 0, do one of the following:
   - To translate zero dates to a date value that is supported by ODBC, select the **Return Minimal Date For Zero Date** check box.
   - To return zero dates as NULL, clear the **Return Minimal Date For Zero Date** check box.
6. To save your settings and close the Simba MySQL ODBC Driver DSN Setup dialog box, click **OK**.

Configuring Additional Connector Options on Windows

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

To configure additional connector options on Windows:

1. To access the additional connector options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then select the **Misc** tab.
2. To ignore spaces between function names and parentheses (`(`) in statements so that function names are treated as keywords, select the **Ignore Space After Function Names** check box.
3. To disable support for transactions, select the **Disable Transaction Support** check box.

4. To translate the minimum ODBC date value to the MySQL zero date value when binding parameters, select the **Bind Minimal Date As Zero Date** check box.

   Note:

   Enabling this option prevents statements from failing because of what seems to be a mismatch in the date values.

5. To emulate prepared statements on the client side, select the **Prepare Statements On The Client** check box.
6. To treat a quotation mark (") as an identifier quote character instead of a string quote character, select the **Enable ANSI Quotes** check box.
7. To save your settings and close the Simba MySQL ODBC Driver DSN Setup 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 MySQL 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 MySQL ODBC Connector, so make sure to disable the feature after you are done using it. To configure logging for the current connection, see Configuring Logging for the Current Connection on page 16.

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.
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. 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.

5. 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.

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

The Simba MySQL ODBC Connector produces the following log files at the location you specify in the Log Path field:

- A `simbaodbcdriver.log` file that logs connector activity that is not specific to a connection.
- A `simbaodbcdriver_connection_[Number].log` file 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 71.

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 Logging for the Current Connection

You can configure logging for the current connection by setting the logging configuration properties in the DSN or in a connection string. For information about the logging configuration properties, see Configuring Logging Options on Windows on page 14. Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

Note:

If the LogLevel configuration property is passed in via the connection string or DSN, the rest of the logging configurations are read from the connection string or DSN and not from the existing connector-wide logging configuration.

To configure logging properties in the DSN, you must modify the Windows registry. For information about the Windows registry, see the Microsoft Windows documentation.

Important:

Editing the Windows Registry incorrectly can potentially cause serious, system-wide problems that may require re-installing Windows to correct.

To add logging configurations to a DSN on Windows:

1. On the Start screen, type **regedit**, and then click the **regedit** search result.
2. Navigate to the appropriate registry key for the bitness of your connector and your machine:
   - 32-bit System DSNs: **HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\ODBC\ODBC.INI*[DSN Name]***
   - 64-bit System DSNs: **HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI*[DSN Name]***
   - 32-bit and 64-bit User DSNs: **HKEY_CURRENT_USER\SOFTWARE\ODBC\ODBC.INI*[DSN Name]***
3. For each configuration option that you want to configure for the current connection, create a value by doing the following:
   a. If the key name value does not already exist, create it. Right-click the *[DSN Name]* and then select **New > String Value**, type the key name of the configuration option, and then press **Enter**.
   b. Right-click the key name and then click **Modify**.

      To confirm the key names for each configuration option, see Connector Configuration Properties on page 51.
   c. In the Edit String dialog box, in the **Value Data** field, type the value for the configuration option.
4. Close the Registry Editor.
5. Restart your ODBC application to make sure that the new settings take effect.

Configuring SSL Connections on Windows

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

Configuring an SSL Connection without Identity Verification

You can configure a connection that is encrypted by 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, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then select the **SSL** tab.

2. From the **SSL Mode** drop-down list, select one of the following options:
   - To use SSL encryption only if the server supports it, select **PREFERRED**.
   - Or, to require SSL encryption for the connection, select **REQUIRED**. If the server does not support SSL, the connection fails.
3. Optionally, in the **SSL Cipher** field, type a comma-separated list of permitted ciphers for encrypting the connection.
4. To specify the minimum version of SSL to use, from the **Minimum TLS** drop-down list, select the minimum version of SSL.
5. To have the connector use the Windows trust store, select the **Use Truststore** checkbox.
6. To save your settings and close the Simba MySQL ODBC Driver DSN Setup dialog box, click **OK**.

Configuring SSL Identity Verification

You can configure one-way verification so that the client verifies the identity of the MySQL server, or you can configure two-way verification so that the client and the sever both verify each other.

In both cases, you must provide a root certificate from a trusted certificate authority (CA) that the connector can use to check the server's certificate. If you are using two-way verification, then you must also provide a certificate that proves the identity of the client and a private key that encrypts the client certificate.

To configure SSL identity verification on Windows:

1. To access the SSL options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then select the **SSL** tab.
2. To specify one or more root certificates from trusted CAs that you want to use to verify the server certificate, do one of the following:
   - To use a specific root certificate, in the **SSL Certificate Authority** field, specify the full path and name of the `.pem` file containing the certificate.
   - Or, to provide multiple root certificates, in the **SSL CA Path** field, specify the full path and name of the directory that contains the certificates. The connector uses the first valid certificate that it finds in the directory.
3. If two-way identity verification is necessary, do the following:
   a. In the **SSL Key** field, specify the full path and name of the file that contains the private key used for encrypting the client certificate.
   b. In the **SSL Certificate** field, specify the full path and name of the `.pem` file containing the certificate used for proving the identity of the client.

4. From the **SSL Mode** drop-down list, select one of the following options:
   - To use SSL encryption and identity verification only if the server supports it, select **VERIFY_CA**.
   - Or, to require SSL encryption and identity verification for the connection, select **VERIFY_IDENTITY**. If the server does not support SSL or if identity verification fails, the connection fails.
5. Optionally, in the **SSL Cipher** field, type a comma-separated list of permitted ciphers for encrypting the connection.
6. To specify the minimum version of SSL to use, from the **Minimum TLS** drop-down list, select the minimum version of SSL.
7. To have the connector use the Windows trust store, select the **Use Truststore** check box.
8. To save your settings and close the Simba MySQL ODBC Driver DSN Setup dialog box, click **OK**.

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 MySQL 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 MySQL.

2. Click the **Drivers** tab and then find the Simba MySQL 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 MySQL ODBC Connector supports MySQL versions 5.7 and 8.0.

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 10.14
  - macOS 10.15
  - macOS 11 (Universal Binary – Intel and ARM support)
- 250MB 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 MySQL ODBC Connector is available for macOS as a `.dmg` file named `Simba MySQL 1.1.dmg`. The connector supports both 32- and 64-bit client applications.

To install the Simba MySQL ODBC Connector on macOS:

1. Double-click **Simba MySQL 1.1.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 files are installed in the `/Library/simba/mysqlodbc` 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 installation directory. You must have root privileges when changing the contents of this folder.

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

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 30.

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 MySQL 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.mysqlodbc
```

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

Linux Driver

For most Linux distributions, you can install the connector using the RPM file or the tarball package. If you are installing the connector on a Debian machine, you must use the Debian package.

Linux System Requirements

The Simba MySQL ODBC Connector supports MySQL versions 5.7 and 8.0.

Install the connectors on client machines where the application is installed. Each client machine that you install the connectors 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 9
  - Ubuntu 16.04 or 18.04
- 90MB 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 connectors, 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 connectors 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.

Additionally, make sure to use the RPM file that has been optimized for your Linux distribution. The following RPM files are available:

- `simbamysql-[Version]-[Release].i686.rpm` for installing the 32-bit connector on Red Hat Enterprise Linux or CentOS machines.
- `simbamysql-[Version]-[Release].x86_64.rpm` for installing the 64-bit connector Red Hat Enterprise Linux or CentOS machines.
- `simbamysql-[Version]-[Release].suse12.i686.rpm` for installing the 32-bit connector on SUSE Linux Enterprise Server machines.
- `simbamysql-[Version]-[Release].suse12.x86_64.rpm` for installing the 64-bit connector on SUSE Linux Enterprise Server machines.

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 connector.

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

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

1. Log in as the root user.
2. Navigate to the folder containing the RPM package for the connector.
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 MySQL ODBC Connector files are installed in the `/opt/simba/mysqlodbc` directory.
4. If you received a license file through email, then copy the license file into the `/opt/simba/mysqlodbc/lib/32` or `/opt/simba/mysqlodbc/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 30.

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 MySQL ODBC Connector is available as a tarball package named `SimbaMySQLODBC-[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  --directory=/opt -zxvf [TarballName]
   ```

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

   The Simba MySQL ODBC Connector files are installed in the `/opt/simba/mysqlodbc` directory.
3. If you received a license file through email, then copy the license file into the `opt/simba/mysqlodbc/lib/32` or `opt/simba/mysqlodbc/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 30.

Installing the Driver on Debian or Ubuntu

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

On 64-bit editions of Debian or Ubuntu, 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 MySQL 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 **mysql_*[Version]-[Release]*_i386.deb** or **mysql_*[Version]-[Release]*_amd64.deb**.
3. Follow the instructions in the installer to complete the installation process.

   The Simba MySQL ODBC Connector files are installed in the `/opt/simba/mysqlodbc` directory.

   > **Note:**
   >
   > If the package manager in your Ubuntu distribution cannot resolve the `libsasl` dependencies automatically when installing the connector, then download and manually install the packages required by the version of the connector that you want to install.

4. If you received a license file via email, then copy the license file into the `/opt/simba/mysqlodbc/lib/32` or `/opt/simba/mysqlodbc/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 30.

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 MySQL 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 or Debian package. Alternatively, you can search the connector's binary file for version number information.

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:
  - `grep -ai driver_version_sb libmysqlodbc_sb[Bitness].so`

    Where *[Bitness]* is either 32 or 64.

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

To verify the connector version number on Linux using the binary file:

1. Navigate to the `/lib` subfolder in your connector installation directory. By default, the path to this directory is: `/opt/simba/mysqlodbc/lib`.
2. Open the connector's `.so` binary file in a text editor, and search for the text `$driver_version_sb$:`. The connector's version number is listed after this text.

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 MySQL 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 30.
- 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 31.

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.mysqlodbc.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 _MYSQL_ODBC_INI to the full path and file name of the `simba.mysqlodbc.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 _MYSQL_ODBC_INI to the full path and file name of the `simba.mysqlodbc.ini` file.

For example, if your `odbc.ini` and `odbcinst.ini` files are located in `/usr/local/odbc` and your `simba.mysqlodbc.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 _MYSQL_ODBC_INI=/etc/simba.mysqlodbc.ini
```

For unixODBC:

```
export ODBCINI=/usr/local/odbc/odbc.ini
export ODBCSYSINI=/usr/local/odbc
export _MYSQL_ODBC_INI=/etc/simba.mysqlodbc.ini
```

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

1. If the _MYSQL_ODBC_INI 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.mysqlodbc.ini`.
3. The connector searches the current working directory of the application for a file named `simba.mysqlodbc.ini`.
4. The connector searches the home directory for a hidden file named `.simba.mysqlodbc.ini` (prefixed with a period).
5. The connector searches the `/etc` directory for a file named `simba.mysqlodbc.ini`.

Configuring ODBC Connections on a Non-Windows Machine

The following sections describe how to configure ODBC connections when using the Simba MySQL 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 36.

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. In the `[ODBC Data Sources]` section, add a new entry by typing a name for the DSN, an equal sign (=), and then the name of the connector.

   For example, on a macOS machine:

   ```
   [ODBC Data Sources]
   Sample DSN=Simba MySQL ODBC Connector
   ```

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

```
[ODBC Data Sources]
Sample DSN=Simba MySQL ODBC Connector 32-bit
```

3. 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/mysqlodbc/lib/libmysqlodbc_
      sbu.dylib
      ```

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

      ```
      Driver=/opt/simba/mysqlodbc/lib/32/libmysqlodbc_
      sb32.so
      ```

   b. Set the `Server` 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:

      ```
      Server=192.168.222.160
      Port=3306
      ```

   c. Set the `UID` property to an appropriate user name for accessing the MySQL database.

      For example:

      ```
      UID=simba
      ```

   d. If you are required to authenticate the connection using the caching_sha2_password, or SHA-256 authentication plugin, then set the `PWD` property to the password corresponding to the user name you specified above.

      For example:

      ```
      PWD=simba123
      ```

   e. Optionally, if you are authenticating the connection using the caching_sha2_password or SHA-256 authentication plugin, specify an RSA public key for encrypting your password by setting the `RSAKey` property to the full

path and name of the file containing the key.

For example:

```
RSAKey=/localhome/simba/authentication/mysql_rsa.pem
```

Note:

If you do not specify an RSA public key or configure SSL for your connection, the connector automatically retrieves a public key from the server and uses it to encrypt your password.

f. If you want to enable encryption and identity verification using SSL, then enable SSL and specify the certificate information. For more information, see Configuring SSL Connections on a Non-Windows Machine on page 38.

g. 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 MySQL ODBC Connector, see Connector Configuration Properties on page 51.

4. 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 31.

For example, the following is an `odbc.ini` configuration file for macOS containing a DSN that connects to MySQL and authenticates the connection:

```
[ODBC Data Sources]
Sample DSN=Simba MySQL ODBC Connector
[Sample DSN]
Driver=/Library/simba/mysqlodbc/lib/libmysqlodbc_sbu.dylib
Server=192.168.222.160
Port=3306
UID=simba
PWD=simba123
```

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 MySQL and authenticates the connection:

```
[ODBC Data Sources]
Sample DSN=Simba MySQL ODBC Connector 32-bit
[Sample DSN]
Driver=/opt/simba/mysqlodbc/lib/32/libmysqlodbc_sb32.so
Server=192.168.222.160
Port=3306
UID=simba
PWD=simba123
```

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/mysqlodbc/lib/libmysqlodbc_
      sbu.dylib
      ```

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

      ```
      Driver=/opt/simba/mysqlodbc/lib/32/libmysqlodbc_
      sb32.so
      ```

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

      For example:

      ```
      Description=Simba MySQL 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 31.

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

```
[ODBC Drivers]
Description= Simba MySQL ODBC Connector
Driver=/Library/simba/mysqlodbc/lib/libmysqlodbc_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/mysqlodbc/lib/32/libmysqlodbc_sb32.so
Description=Simba MySQL ODBC Connector(64 bit)
Driver=/opt/simba/mysqlodbc/lib/64/libmysqlodbc_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 SSL connections, see Configuring SSL Connections 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 51.

Configuring SSL Connections on a Non-Windows Machine

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

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 an SSL Connection without Identity Verification

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

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

1. Enable SSL encryption by doing one of the following:
   - To use SSL encryption only if the server supports it, set the `SSLMode` property to `1`.
   - Or, to require SSL encryption for the connection, set the `SSLMode` property to `2`. If the server does not support SSL, the connection fails.
2. Optionally, set the `SSLCipher` property to a comma-separated list of permitted ciphers for encrypting the connection.
3. To specify the minimum version of SSL to use, set the `Min_TLS` property to the minimum version of SSL. Supported options include `1.0` for TLS 1.0, `1.1` for TLS 1.1, and `1.2` for TLS 1.2.

Configuring SSL Identity Verification

You can configure one-way verification so that the client verifies the identity of the MySQL server, or you can configure two-way verification so that the client and the sever both verify each other.

In both cases, you must provide a root certificate from a trusted certificate authority (CA) that the connector can use to check the server's certificate. If you are using two-way verification, then you must also provide a certificate that proves the identity of the client and a private key that encrypts the client certificate.

To configure SSL identity verification on a non-Windows machine:

1. Enable SSL encryption by doing one of the following:
    - To use SSL encryption and identity verification only if the server supports it, set the `SSLMode` property to `3`.
    - Or, to require SSL encryption and identity verification for the connection, set the `SSLMode` property to `4`. If the server does not support SSL or if identity verification fails, the connection fails.
2. To specify one or more root certificates from trusted CAs that you want to use to verify the server certificate, do one of the following:
    - To use a specific root certificate, set the `SSLCA` property to the full path and name of the `.pem` file containing the certificate.
    - Or, to provide multiple root certificates, set the `SSLCAPath` property to the full path and name of the directory that contains the certificates. The connector uses the first valid certificate that it finds in the directory.
3. If two-way identity verification is necessary, do the following:
    a. Set the `SSLCert` property to the full path and name of the `.pem` file containing the certificate used for proving the identity of the client.
    b. Set the `SSLKey` property to the full path and name of the file that contains the private key used for encrypting the client certificate.
4. Optionally, set the `SSLCipher` property to a comma-separated list of permitted ciphers for encrypting the connection.
5. To specify the minimum version of SSL to use, set the `Min_TLS` property to the minimum version of SSL. Supported options include `1.0` for TLS 1.0, `1.1` for TLS 1.1, and `1.2` for TLS 1.2.

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.

You can set the connection properties described below in a connection string, in a DSN (in the `odbc.ini` file), or as a connector-wide setting (in the `simba.mysqlodbc.ini` file). Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

To enable logging on a non-Windows machine:

1. 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.

2. Set the `LogPath` key to the full path to the folder where you want to save log files.
3. 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.

4. 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.

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

The Simba MySQL ODBC Connector produces the following log files at the location you specify using the `LogPath` key:

- A `mysqlodbcdriver.log` file that logs connector activity that is not specific to a connection.
- A `mysqlodbcdriver_connection_[Number].log` file 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 71.

To disable logging on a non-Windows machine:

1. Set the `LogLevel` key to `0`.
2. Save the `simba.mysqlodbc.ini` configuration file.
3. 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 51.

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.

Important:

When you connect to the data store using a DSN-less connection string, the connector does not encrypt your credentials.

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

- *[PortNumber]* is the number of the TCP port that the MySQL server uses to listen for client connections.
- *[ServerInfo]* is the IP address or host name of the MySQL server to which you are connecting.

- *[YourPassword]* is the password corresponding to your user name.
- *[YourUserName]* is the user name that you use to access the MySQL server.

Connecting to a MySQL Server that Uses the No Authentication

The following is the format of a DSN-less connection string for connecting to a MySQL server that does not require authentication. You must always specify a user name when connecting to MySQL, so this same connection string format can be used for both types of connections.

```
Driver=Simba MySQL ODBC Driver;Server=[ServerInfo];
Port=[PortNumber];UID=[YourUserName]
```

For example:

```
Driver=Simba MySQL ODBC Driver;Server=192.168.222.160;
Port=3306;UID=simba
```

Connecting to a MySQL Server that Uses the caching_sha2_password, or SHA-256 Authentication Plugin

The following is the format of a DSN-less connection string for connecting to a MySQL server that uses either the caching_sha2_password, or SHA-256 authentication plugin:

```
Driver=Simba MySQL ODBC Driver;Server=[ServerInfo];
Port=[PortNumber];UID=[YourUserName];PWD=[YourPassword]
```

For example:

```
Driver=Simba MySQL ODBC Driver;Server=192.168.222.160;
Port=3306;UID=simba;PWD=simba123
```

Features

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

Data Types

The Simba MySQL ODBC Connector supports many common data formats, converting between MySQL data types and SQL data types.

The table below lists the supported data type mappings.

MySQL Type	SQL Type
BIGINT BIGINT UNSIGNED	SQL_BIGINT **Note:** SQL_INTEGER is returned instead if the Treat BIGINT Columns As INT Columns option (the `No_BIGINT` property) is enabled.
BINARY	SQL_BINARY
BIT (M)	• SQL_BIT when M = 1 • SQL_BINARY when M > 1
BLOB	SQL_LONGVARBINARY
BOOL BOOLEAN	SQL_TINYINT
CHAR	SQL_WCHAR

MySQL Type	SQL Type
DATE	• SQL_TYPE_DATE if the application uses ODBC version 3.00 or later. • SQL_DATE if the application uses an ODBC version earlier than 3.00.
DATETIME	• SQL_TYPE_TIMESTAMP if the application uses ODBC version 3.00 or later. • SQL_TIMESTAMP if the application uses an ODBC version earlier than 3.00.
DEC DECIMAL	SQL_DECIMAL
DOUBLE DOUBLE PRECISION	SQL_DOUBLE
ENUM	SQL_WCHAR
FLOAT	SQL_REAL
GEOMETRY	SQL_LONGVARBINARY
GEOMETRYCOLLECTION	SQL_LONGVARBINARY
INT INTEGER INTEGER UNSIGNED	SQL_INTEGER

MySQL Type	SQL Type
JSON **Note:** Only supported in MySQL 5.7 or later.	SQL_BINARY
LINESTRING	SQL_LONGVARBINARY
LONGBLOB	SQL_LONGVARBINARY
LONGTEXT	SQL_WLONGVARCHAR
MEDIUMBLOB	SQL_LONGVARBINARY
MEDIUMINT MEDIUMINT UNSIGNED	SQL_INTEGER
MEDIUMTEXT	SQL_WLONGVARCHAR
MULTIPOINT MULTILINESTRING MULTIPOLYGON	SQL_LONGVARBINARY
NUMERIC	SQL_DECIMAL
POINT	SQL_LONGVARBINARY
POLYGON	SQL_LONGVARBINARY
SET	SQL_WCHAR
SMALLINT SMALLINT UNSIGNED	SQL_SMALLINT
TEXT	SQL_WLONGVARCHAR

MySQL Type	SQL Type
TIME	• SQL_TYPE_TIME if the application uses ODBC version 3.00 or later. • SQL_TIME if the application uses an ODBC version earlier than 3.00.
TIMESTAMP	• SQL_TYPE_TIMESTAMP if the application uses ODBC version 3.00 or later. • SQL_TIMESTAMP if the application uses an ODBC version earlier than 3.00.
TINYBLOB	SQL_LONGVARBINARY
TINYINT TINYINT UNSIGNED	SQL_TINYINT
TINYTEXT	SQL_WLONGVARCHAR
VARBINARY	SQL_VARBINARY
VARCHAR	SQL_WVARCHAR
YEAR	SQL_SMALLINT

Security and Authentication

To protect data from unauthorized access, some MySQL data stores require connections to be authenticated with user credentials or encrypted using the SSL protocol. The Simba MySQL 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 . The SSL version used for the connection is the highest version that is supported by both the connector and the server.

Note:

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

By default, the connector supports TLS 1.0, 1.1, and 1.2. You can configure the connector to use a specific TLS version. For more information, see Configuring SSL Connections on Windows on page 17 or Configuring SSL Connections on a Non-Windows Machine on page 38.

The connector provides a mechanism that enables you to authenticate your connection using the caching_sha2_password, SHA-256, or Windows native authentication plugins. The settings in the MySQL server determine which plugin is used and what credentials you need to provide. The default plugin is caching_sha2_password. For detailed connector configuration instructions, see Creating a Data Source Name on Windows on page 8 or Creating a Data Source Name on a Non-Windows Machine on page 33.

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

- No identity verification
- One-way authentication
- 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 Connections on Windows on page 17 or Configuring SSL Connections on a Non-Windows Machine on page 38.

Connector Configuration Properties

Connector Configuration Options lists the configuration options available in the Simba MySQL 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 MySQL ODBC Driver DSN Setup
- 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 MySQL ODBC Connector, or via the key name when using a connection string or configuring a connection from a Linux or macOS machine:

Allow Multiple Statements

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

Description

This option specifies whether the connector supports batched statements.

- Enabled (`1`): The connector allows multiple statements to be processed in a single execution.
- Disabled (`0`): The connector does not allow batched statements.

Always Handle Binary Function Results As Character Data

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

Description

This option specifies whether the connector returns Binary columns as Character columns when returning function results.

- Enabled (`1`): The connector returns Binary columns as Character columns.
- Disabled (`0`): The connector returns Binary columns normally.

Bind Minimal Date As Zero Date

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

Description

This option specifies whether the connector translates the minimum ODBC date value (`0000-01-01`) to the zero date value used in MySQL (`0000-00-00`) when binding parameters. Enabling this option prevents statements from failing because of what seems to be a mismatch in the date values.

- Enabled (`1`): The connector translates the minimum ODBC date value to the MySQL zero date value when binding parameters.
- Disabled (`0`): The connector does not modify the date values.

Can Handle Expired Password

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

Description

This option determines how the connector responds when you attempt to connect to the server using an expired password.

- Enabled (`1`): The connector establishes a sandboxed connection through which you can issue a SET PASSWORD statement to update the password.
- Disabled (`0`): The connector returns an error, and the connection fails.

Catalog

Key Name	Default Value	Required
`Database`	None	No

Description

The name of the MySQL database to connect to by default. You can still issue queries on other databases by specifying the database schema in your query statement.

Disable Transaction Support

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

Description

This option specifies whether the connector supports transactions.

- Enabled (`1`): The connector does not support transactions.
- Disabled (`0`): The connector supports transactions.

Don't Cache Results Of Forward-Only Cursors

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

Description

This option specifies whether the connector caches result sets into memory. Enable this option to reduce memory consumption when working with large result sets.

- Enabled (`1`): The connector does not cache result sets into memory.
- Disabled (`0`): The connector caches result sets into memory.

Don't Prompt When Connecting

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

Description

This option specifies whether the connector allows dialog boxes to open and prompt you for more information during connection time.

- Enabled (`1`): The connector does not allow dialog boxes to open.
- Disabled (`0`): The connector allows dialog boxes to open and prompt you for more connection information, if necessary.

Enable ANSI Quotes

Key Name	Default Value	Required
`ANSI_Quotes`	Clear (`false`)	No

Description

This option specifies whether the connector treats a quotation mark (") as an identifier quote character or a string quote character.

- Enabled (`true`): The connector treats a quotation mark (") as an identifier quote character.

  When the connector connects to the data store, it executes the following statement:

  SET SESSION sql_mode = 'ANSI_QUOTES'.
- Disabled (`false`): The connector treats a quotation mark as a string quote character.

Enable Automatic Reconnect

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

Description

This option specifies whether the connector attempts to automatically reconnect to the server when a communication link error occurs.

Important:

Do not enable this option if you are working with transactions. Auto-reconnecting during an incomplete transaction may corrupt the data.

- Enabled (`1`): The connector attempts to reconnect.
- Disabled (`0`): The connector does not attempt to reconnect.

Enable SQL_AUTO_IS_NULL

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

Description

This option specifies whether the connector modifies the `sql_auto_is_null` setting in the server. For more information, see "Server System Variables" in the *MySQL Reference Manual*: https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_sql_auto_is_null.

- Enabled (`1`): The connector does not modify the `sql_auto_is_null` setting in the server, and supports `sql_auto_is_null=1` if it is specified in the server.
- Disabled (`0`): The connector sets the `sql_auto_is_null` server system variable to `0`, and does not support the retrieval of `AUTO_INCREMENT` values using IS NULL queries.

Enable Table Types

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

Description

This option specifies whether the connector recognizes table type information from the data source. By default, the connector only recognizes a single, generic table type.

- Enabled (`1`): The connector recognizes the following table types: TABLE and SYSTEM TABLE.
- Disabled (`0`): All tables returned from the data source have the generic type TABLE.

Host

Key Name	Default Value	Required
`Server`	`localhost`	Yes

Description

The host name or IP address of the MySQL server.

Ignore Schema In Column Specifications

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

Description

This option specifies whether the connector accepts column names that include the database name (using the format *[Database].[Table].[Column]*).

- Enabled (`1`): The connector ignores the database name.
- Disabled (`0`): The connector returns an error if a column name specified in a statement includes the database name.

Ignore Space After Function Names

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

Description

This option specifies whether the connector ignores spaces between function names and parentheses (`(`) so that function names are treated as keywords. Enabling this option provides compatibility with applications such as PowerBuilder.

- Enabled (`1`): The connector ignores the spaces, and treats function names as keywords.
- Disabled (`0`): The connector does not ignore the spaces.

Include Table Name In SQLDescribeCol()

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

Description

This option specifies whether the connector returns fully-qualified column names when the `SQLDescribeCol()` function is called.

- Enabled (`1`): The connector returns fully-qualified column names.
- Disabled (`0`): The connector returns column names that do not include the table name.

Initial Statement

Key Name	Default Value	Required
`InitStmt`	None	No

Description

A statement that the connector executes immediately after connecting to the server.

Interactive Client

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

Description

This option specifies whether the connector manages the connection using the `interactive_timeout` value specified in the server.

- Enabled (`1`): The connector closes the connection if the time interval specified by `interactive_timeout` passes without any activity occurring.
- Disabled (`0`): The connector does not close inactive connections.

Limit Column Size To A Signed 32-bit Value

Key Name	Default Value	Required
`COLUMN_SIZE_S32`	Clear (`0`)	No

Description

This option specifies whether the connector limits column sizes to 32-bit values, preventing problems with larger column sizes in applications that do not support them.

- `0`: The connector uses 32-bit and 64-bit column sizes as required the by data.
- `1`: The connector limits column sizes to signed 32-bit values.

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.
- When logging with connection strings and DSNs, this option only applies to per-connection logs.

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:

- A `mysqlodbcdriver.log` file that logs connector activity that is not specific to a connection.
- A `mysqlodbcdriver_connection_[Number].log` file 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 71.

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:**
>
> When logging with connection strings and DSNs, this option only applies to per-connection logs.

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:**
>
> When logging with connection strings and DSNs, this option only applies to per-connection logs.

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:

When logging with connection strings and DSNs, this option only applies to per-connection logs.

Minimum TLS

Key Name	Default Value	Required
`Min_TLS`	TLS 1.2 (`1.2`)	No

Description

The minimum version of TLS/SSL that the connector allows the data store to use for encrypting connections. For example, if TLS 1.1 is specified, TLS 1.0 cannot be used to encrypt connections.

- TLS 1.0 (`1.0`): The connection must use at least TLS 1.0.
- TLS 1.1 (`1.1`): The connection must use at least TLS 1.1.
- TLS 1.2 (`1.2`): The connection must use at least TLS 1.2.

Password

Key Name	Default Value	Required

`PWD`
OR
`Password`
	None	Yes, if connecting to a server that uses the caching_sha2_ password, or SHA-256 authentication plugins.

Description

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

Port

Key Name	Default Value	Required
`Port`	`3306`	Yes

Description

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

Prepare Statements On The Client

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

Description

This option specifies whether the connector emulates prepared statements on the client side. This functionality is required when using prepared statements for certain types of queries. If the connector fails to execute your prepared statement, enable this option and then try executing the statement again.

- Enabled (`1`): The connector emulates prepared statements on the client side.
- Disabled (`0`): The connector does not emulate prepared statements on the client side.

Return Matched Rows Instead Of Affected Rows

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

Description

This option specifies whether the connector returns the number of matched rows or the number of affected rows after a DML operation.

- Enabled (`1`): The connector returns the number of matched rows.
- Disabled (`0`): The connector returns the number of affected rows.

Return Minimal Date For Zero Date

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

Description

This option specifies whether the connector translates zero dates (where the month or day is `0`) to a date value that is supported by ODBC. For example, when this option is enabled, the date (`0000-00-00`) is translated to (`0000-01-01`).

- Enabled (`1`): The connector translates the zero dates to a date value that is supported by ODBC.
- Disabled (`0`): The connector returns zero dates as NULL.

RSA Public Key

Key Name	Default Value	Required
`RSAKey`	None	No

Description

The full path and name of a file containing an RSA public key for encrypting your password.

This option is applicable only when you connect to a server that uses the caching_sha2_password or SHA-256 authentication plugin.

Return Default metadata for SQLDescribeParam

Key Name	Default Value	Required
RETURN_DEFAULT_METADATA_FOR_SQLDESCRIBECOLUMNS	0	No

Description

This option specifies whether the connector returns default metadata as SQL_VARCHAR or an error, if the actual type of the parameter cannot be determined.

- Enabled (1): The connector returns default metadata for SQLDescribeParam.
- Disabled (0): The connector returns an error, if it fails to determine the appropriate type.

SSL CA Path

Key Name	Default Value	Required
SSLCAPath	None	Yes, if using SSL identity verification and the SSL Certificate Authority option (the SSLCA property) is not set.

Description

The full path and name of the directory that contains root certificates from trusted CAs. The certificates must be stored as .pem files.

Use this option to specify root certificates for verifying the server certificate during an SSL identity verification. The connector uses the first valid certificate that it finds in the directory.

To use a specific root certificate, configure the SSL Certificate Authority option (the SSLCA property) instead. For more information, see SSL Certificate Authority on page 65.

SSL Certificate

Key Name	Default Value	Required
`SSLCert`	None	Yes, if using two-way SSL verification.

Description

The full path and name of a `.pem` file containing the SSL certificate used for proving the identity of the client.

To specify a private key for encrypting this certificate before sending it to the server, use the SSL Key option (the `SSLKey` property). For more information, see SSL Key on page 66.

SSL Certificate Authority

Key Name	Default Value	Required
`SSLCA`	None	Yes, if using SSL identity verification and the SSL CA Path option (the `SSLCAPath` property) is not set.

Description

The full path and name of a `.pem` file containing a root certificate from a trusted CA.

Use this option to specify a root certificate for verifying the server certificate during SSL identity verification.

SSL Cipher

Key Name	Default Value	Required
`SSLCipher`	None	No

Description

A comma-separated list of permitted ciphers for the SSL connection.

If you do not set this option, the connector automatically selects an appropriate cipher for the connection.

SSL Key

Key Name	Default Value	Required
SSLKey	None	Yes, if using two-way SSL verification.

Description

The full path and name of a file containing the private key used for encrypting the client-side certificate during two-way SSL verification.

The client-side certificate is specified using the SSL Certificate option (the SSLCert property). For more information, see SSL Certificate on page 65.

SSL Mode

Key Name	Default Value	Required
SSLMode	PREFERRED (1)	No

Description

This option specifies whether the connector uses SSL encryption and verification when connecting to MySQL.

- **DISABLED** (0): SSL is not used.
- **PREFERRED** (1): Use SSL encryption if the server supports it. Otherwise, connect without using SSL.
- **REQUIRED** (2): Use SSL encryption. If the server does not support SSL, the connection fails.
- **VERIFY_CA** (3): Use SSL encryption and either one-way or two-way identity verification. If the server does not support SSL or if identity verification fails, the connection fails.
- **VERIFY_IDENTITY** (4): Use SSL encryption and either one-way or two-way identity verification, and also verify the host name of the server. If the server does not support SSL or if a certificate or host name cannot be verified, the connection fails.

Treat BIGINT Columns As INT Columns

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

Description

This option specifies whether the connector returns BIGINT data as SQL_BIGINT or SQL_INTEGER data.

- Enabled (`1`): The connector returns BIGINT data as SQL_INTEGER data.
- Disabled (`0`): The connector returns BIGINT data as SQL_BIGINT data.

TrustedCertificates

Key Name	Default Value	Required
`TrustedCerts`	The `cacerts.pem` file in the `\lib` subfolder within the connector's installation directory. The exact file path varies depending on the version of the connector that is installed. For example, the path for the Windows connector is different from the path for the macOS connector.	No

Description

The full path of the `.pem` file containing trusted CA certificates, for verifying the server.

If this option is not set, then the connector defaults to using the trusted CA certificates `.pem` file installed by the connector. To use the trusted CA certificates in the `.pem` file, set the `UseSystemTrustStore` property to `0` or clear the Use System Trust Store check box in the SSL Options dialog.

Use Compression

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

Description

This option specifies whether the connector compresses the communication between the client and the server. Compression may reduce the amount of network traffic for certain workflows, but in some cases it may increase CPU usage.

- Enabled (`1`): The connector compresses the communication between the client and the server.
- Disabled (`0`): The connector does not use compression.

Use Trust Store

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

Description

This option specifies whether to use a CA certificate from the system trust store, or from a specified `.pem` file.

- Enabled (`1`): The connector verifies the connection using a certificate in the system trust store.
- Disabled (`0`): The connector verifies the connection using a specified `.pem` file. For information about specifying a `.pem` file, see .

Note:

This option is only available on Windows.

User

Key Name	Default Value	Required

`UID`
OR
`User`
	None	Yes

Description

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

Configuration Options Having Only Key Names

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

- Driver on page 69
- MaxCatalogNameLen on page 70
- MaxColumnNameLen on page 70
- MaxSchemaNameLen on page 70
- MaxTableNameLen on page 71

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

- UseLogPrefix on page 71

Driver

Key Name	Default Value	Required
`Driver`	`MySQL 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 (`MySQL 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.

MaxCatalogNameLen

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

Description

The maximum number of characters that can be returned for catalog names.

This option can be set to any integer from 0 to 65535, inclusive. To indicate that there is no maximum length or that the length is unknown, set this option to `0`.

MaxColumnNameLen

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

Description

The maximum number of characters that can be returned for column names.

This option can be set to any integer from 0 to 65535, inclusive. To indicate that there is no maximum length or that the length is unknown, set this option to `0`.

MaxSchemaNameLen

Key Name	Default Value	Required
`MaxSchemaNameLen`	`256`	No

Description

The maximum number of characters that can be returned for schema names.

This option can be set to any integer from 0 to 65535, inclusive. To indicate that there is no maximum length or that the length is unknown, set this option to `0`.

MaxTableNameLen

Key Name	Default Value	Required
MaxTableNameLen	0	No

Description

The maximum number of characters that can be returned for table names.

This option can be set to any integer from 0 to 65535, inclusive. To indicate that there is no maximum length or that the length is unknown, set this option to 0.

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_simbamysqlodbcdriver.log and jdoe_7836_simbamysqlodbcdriver_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 MySQL ODBC Connector\Driver**
- Otherwise: **HKEY_LOCAL_MACHINE\SOFTWARE\Simba\Simba MySQL 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.mysqlodbc.ini` file.

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.

MySQL is a registered trademark of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

All other trademarks are trademarks of their respective owners.