insightsoftware

Simba SQL Server ODBC Data Connector

Installation and Configuration Guide

Version 1.5.30

March 2025

Contents

Copyright

Contact Us

www.insightsoftware.com

About This Guide

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

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

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

- Familiarity with the platform on which you are using the Simba SQL Server ODBC Connector

- Ability to use the data source to which the Simba SQL Server 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 is 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.

About the Simba SQL Server ODBC Connector-1.5.30

The Simba SQL Server ODBC Connector enables Business Intelligence (BI), analytics, and reporting on data that is stored in Microsoft SQL Server 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: https://insightsoftware.com/blog/what-is-odbc/. For complete information about the ODBC specification, see the *ODBC API Reference* from the Microsoft documentation: https://docs.microsoft.com/en-us/sql/odbc/reference/syntax/odbc-api-reference.

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

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

Windows Connector

This section provides an overview of the Connector in the Windows platform, outlining the required system specifications and the steps for installing and configuring the connector in Windows environments.

Windows System Requirements

The Simba SQL Server ODBC Connector supports SQL Server versions 2016 to 2022.

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 2022, 2019, 2016, or 2012
    - 100 MB of available disk space

Before the connector can be used, the Visual C++ Redistributable for Visual Studio 2015-2022 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 Connector in Windows

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 Connector 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 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:

- `SimbaSQLServerODBC32.msi` for 32-bit applications

- `SimbaSQLServerODBC64.msi` for 64-bit applications

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

To install the Simba SQL Server ODBC Connector in Windows:

1. Depending on the bitness of your client application, double-click to run **SimbaSQLServerODBC32.msi** or **SimbaSQLServerODBC64.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.

Verifying the Connector Version Number in Windows

If you need to verify the version of the Simba SQL Server 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 in Windows:

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

    > ℹ **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 SQL Server.

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

Creating a Data Source Name in Windows

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

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

To create a Data Source Name in Windows:

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

    > ℹ **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 SQL Server.

2. In the ODBC Data Source Administrator, click the **Drivers** tab, and then scroll down as needed to confirm that the Simba SQL Server ODBC Connector appears in the alphabetical list of ODBC drivers that are installed on your system.

3. Choose one:

- To create a DSN that only the user currently logged into Windows can use, click the **User DSN** tab.

- Or, to create a DSN that all users who log into Windows can use, click the **System DSN** tab.

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

4. Click **Add**.

5. In the Create New Data Source dialog box, select **Simba SQL Server ODBC Connector** and then click **Finish**. The Simba SQL Server 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 **Server** field, type the name or IP address of the host where your SQL Server instance is running.

> **Note:** You can specify the name of the SQL Server instance by using the syntax *[ServerInfo]\[Instance]*, where *[ServerInfo]* is the IP address or host name of the server and *[Instance]* is the name of the instance. Make sure that the **Port** field is left empty in this scenario.

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

> **Note:** The default port used by SQL Server is 1433.

10. In the **Database** field, type the name of the database that you want to access.

11. Optionally, in the **Application Name** field, type the name of the application that calls SQLDriverConnect.

12. To use TLS to encrypt all communication with the SQL Server instance before sending it over the network, select the **Encrypt** check box and then do one of the following:

- To enable one-way authentication so that the connector verifies the server certificate using a CA certificate, in the **CA Certificate** field, type the full path and file name of the CA certificate that you want to use.

- Or, to trust the server certificate instead of verifying it, select the **Trust Server Certificate** check box.

13. To specify the minimum version of TLS that the connector must use to connect to the server, from the **Minimum TLS/SSL** drop-down list, select the required TLS version.

14. To configure the connector to enable read-only mode, from the **Application Intent** drop-down list, select the ReadOnly option. For more information, see Application Intent.

15. Configure authentication by specifying the authentication mechanism. For more information, see Configuring Authentication in Windows.

16. To return certain SQL Server-specific data types as ODBC data types instead of SQL Server data types, select the **Return SQL Server-Specific Types as ODBC Types** check box. For a complete list of the data types that this setting applies to, see Return SQL Server-Specific Types as ODBC Types.

17. To include temporary tables in the results when calling SQLTables, select the **Enable Table Types** check box.

18. To optimize connectivity for Always On Availability Groups and Failover Cluster Instances, select the **Multi-Subnet Failover** check box.

19. To configure logging behavior for the connector, click **Logging Options**. For more information, see Configuring Connector-wide Logging Options in Windows.

20. To test the connection, click **Test**. Review the results as needed, and then click **OK**.

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

21. To save your settings and close the Simba SQL Server ODBC Driver DSN Setup dialog box, click **OK**.

22. To close the ODBC Data Source Administrator, click **OK**.

Configuring Authentication in Windows

Connections to SQL Server require authentication. You can authenticate the connection using your SQL Server user account, the Kerberos protocol, or the NTLM protocol.

Using a User Account

You can authenticate the connection by providing your user name and password for accessing the SQL Server instance.

To configure authentication using a user account:

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

2. In the **User** and **Password** fields, type your credentials for accessing the server.

Using Kerberos

You can authenticate the connection by using the Kerberos protocol.

To configure authentication using Kerberos:

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

2. Select the **Use Trusted Connection** check box.

3. Optionally, in the **Server SPN** field, type the service principal name of your SQL Server instance.

> **Note:** If you leave the field empty, then the connector uses **MSSQLSvc/[HostName]**: **[Port]** as the service principal name, where *[HostName]* is the IP address or host name of the server and *[Port]* is the number of the port that you are connecting to.

Using NTLM

You can authenticate the connection by using the NTLM protocol.

To configure authentication using NTLM:

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

2. Select the **Use NTLM** check box.

> **Important:** Make sure that you clear the **Use Trusted Connection** check box. If **Use Trusted Connection** is selected, then the **Use NTLM** option is not available and the connector uses Kerberos authentication instead.

3. In the **User** and **Password** fields, type your credentials for accessing the server.

Configuring Connector-wide Logging Options in Windows

To help troubleshoot issues, you can enable logging in the connector or in the wire protocol component. In addition to these forms of logging supported by the Simba SQL Server 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.

The settings for logging apply to every connection that uses the Simba SQL Server 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.

Connector Logging

Use connector logging to track the activity in the Simba SQL Server ODBC Connector. You can specify the amount of detail included in the log files. The table below lists the logging levels that are available, in order from least verbose to most verbose.

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.

Logging Level	Description
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.

To enable connector logging in 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. In the **Log Level** drop-down list, select the desired level of information to include in log files.

3. In the **Log Path** field, type the full path to the folder where you want to save log files.

4. In the **Max Number Files** field, type the maximum number of log files to keep.

   > ℹ️ **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).

   > ℹ️ **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 SQL Server ODBC Connector produces the following log files at the location you specify using the `LogPath` key:

- A `simbasqlserverodbcdriver.log` file that logs connector activity that is not specific to a connection.

- A `simbasqlserverodbcdriver_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.

To disable connector logging in 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. In the **Log Level** drop-down list, select **LOG_OFF**.

3. Click **OK**.

Wire Protocol Component Logging

Use wire protocol component logging to track the data activity between the connector and the SQL Server instance. You can specify the amount of detail to include in the log files. The table below lists the logging levels that are available, in order from least verbose to most verbose.

Logging Level	Description
0	Disables all logging.
1	Logs error events that might allow the wire protocol component to continue running. 1 is the default logging level.
2	Logs general information that describes the progress of the wire protocol component.
3	Logs detailed information that is useful for debugging the wire protocol component.
4	Logs all activity in the wire protocol component.

Wire protocol component logging is configured through the TDSTRACE environment variable.

To enable wire protocol component logging in Windows:

1. Open the System Information dialog box:

   - If you are using Windows 7 or earlier, click **Start** , then right-click **Computer**, and then click **Properties**.

   - Or, if you are using Windows 8 or later, on the Start screen, right-click **This PC** and then click **Properties**.

2. Click **Advanced System Settings**.

3. In the System Properties dialog box, click the **Advanced** tab and then click **Environment Variables**.

4. Choose one:

   - If the TDSTRACE variable already exists in the System Variables list, select it and then click **Edit**.

   - Or, if the TDSTRACE variable does not appear in the System Variables list, click **New** and then in the **Variable Name** field type **TDSTRACE**.

5. In the **Variable Value** field, type *[LoggingLevel]:[LogFilePath]*, where *[LoggingLevel]* is the logging level indicating the amount of detail to include in the log file and *[LogFilePath]* is the full path of the log file.

   For example, the value **3:C:\Logs\MyWireLog.log** configures the wire protocol component to log debugging information in a file named `MyWireLog.log` located in the `C:\Logs` folder.

6. To save your changes and close the Edit System Variable dialog box, click **OK**

7. Click **OK** to close the Environment Variables dialog box, and then click **OK** to close the System Properties dialog box.

To disable wire protocol component logging in Windows:

1. Open the System Information dialog box:

   - If you are using Windows 7 or earlier, click **Start** , then right-click **Computer**, and then click **Properties**.

   - Or, if you are using Windows 8 or later, on the Start screen, right-click **This PC** and then click **Properties**.

2. Click **Advanced System Settings**.

3. In the System Properties dialog box, click the **Advanced** tab and then click **Environment Variables**.

4. Select **TDSTRACE** from the System Variables list and then click **Edit**.

5. In the **Variable Value** field, replace the existing value with **0**.

6. To save your changes and close the Edit System Variable dialog box, click **OK**.

7. Click **OK** to close the Environment Variables dialog box, and then click **OK** to close the System Properties dialog box.

ODBC Tracing

You can use the ODBC Data Source Administrator to trace connection activity in the ODBC layer.

To start tracing using the ODBC Data Source Administrator:

1. In the ODBC Data Source Administrator, click the **Tracing** tab.

2. In the Log File Path area, click **Browse**. In the Select ODBC Log File dialog box, browse to the location where you want to save the log file, then type a descriptive file name in the **File** name field, and then click **Save**.

3. On the Tracing tab, click **Start Tracing Now**.

To stop ODBC Data Source Administrator tracing:

- In the ODBC Data Source Administrator, on the Tracing tab, click **Stop Tracing Now**.

For more information about tracing using the ODBC Data Source Administrator, see "How to Generate an ODBC Trace with ODBC Data Source Administrator" on the Microsoft Support website: http://support.microsoft.com/kb/274551.

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 Connector-wide Logging Options in Windows. Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

> ℹ **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.

To add logging configurations to a DSN in 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. Navigate to the appropriate registry key for the bitness of your connector and your machine:

4. 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 Options.
   c. In the Edit String dialog box, in the **Value Data** field, type the value for the configuration option.

5. Close the Registry Editor.

6. Restart your ODBC application to make sure that the new settings take effect.

macOS Connector

This section provides an overview of the Connector in the mac OS platform, outlining the required system specifications and the steps for installing and configuring the connector in mac OS environments.

macOS System Requirements

The Simba SQL Server ODBC Connector supports SQL Server versions 2016 to 2022.

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 ARM

- 150MB 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 Connector in 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 Connectors Installation Guide*.

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

To install the Simba SQL Server ODBC Connector in macOS:

1. Double-click **SimbaSQLServerODBC.dmg** to mount the disk image.

2. Double-click **SimbaSQLServerODBC.dmg** to mount the disk image.

3. In the installer, click **Continue**.

4. 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.

5. Optionally, to change the installation location, click **Change Install Location**, then select the desired location, and then click **Continue**.

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

6. To accept the installation location and begin the installation, click **Install**.

7. When the installation completes, click **Close**.

8. If you received a license file through email, then copy the license file into the `/lib` subfolder in the connector installation directory. You must have root privileges when changing the contents of this folder.

    For example, if you installed the connector to the default location, you would copy the license file into the `/Library/simba/sqlserverodbc/lib` 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 in Non-Windows Machines

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

Verifying the Connector Version Number in macOS

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

To verify the connector version number in macOS:

- At the Terminal, run the command:

```
pkgutil --info com.simba.sqlserverodbc
```

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

Linux Connector

This section provides an overview of the Connector in the Linux platform, outlining the required system specifications and the steps for installing and configuring the connector in Linux environments.

Linux System Requirements

The Simba SQL Server ODBC Connector supports SQL Server versions 2016 to 2022.

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 distributions:
  - Red Hat® Enterprise Linux® (RHEL) 8
  - CentOS 7 or 8
  - SUSE Linux Enterprise Server (SLES) 15
  - Debian 9
  - Ubuntu 16.04, 18.04, or 24.04

- 150MB 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 connector, you must have root access on the machine.

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 SQL Server ODBC Connector is available as a tarball package named , 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 SQL Server ODBC Connector files are installed in the `/opt/simba/sqlserverodbc` directory.

3. If you received a license file through email, then copy the license file into the `opt/simba/sqlserverodbc/lib/32` or `opt/simba/sqlserverodbc/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 in Non-Windows Machines.

Verifying the Connector Version Number in Linux

If you need to verify the version of the Simba SQL Server ODBC Connector that is installed on your Linux machine, you can search the connector's binary file for version number information. Alternatively, you can search the connector's binary file for version number information.

To verify the connector version number in 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/sqlserverodbc`.

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 in Non-Windows Machines

To make sure that the ODBC Driver manager on your machine is configured to work with the Simba SQL Server 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 in Non-Windows Machines .

- 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 Connector Configuration Files.

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

Specifying ODBC Driver Managers in 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 Connector 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.sqlserverodbc.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 SIMBASQLSERVERINI to the full path and file name of the `simba.sqlserverodbc.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 SIMBASQLSERVERINI to the full path and file name of the `simba.sqlserverodbc.ini` file.

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

For unixODBC:

```
export ODBCINI=/usr/local/odbc/odbc.ini

export ODBCSYSINI=/usr/local/odbc

export SIMBASQLSERVERINI=/etc/simba.sqlserverodbc.ini
```

To locate the `simba.sqlserverodbc.ini` file, the connector uses the following search order:

1. If the SIMBASQLSERVERINI 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.sqlserverodbc.ini`.

3. The connector searches the current working directory of the application for a file named `simba.sqlserverodbc.ini`.

4. The connector searches the home directory for a hidden file named `.simba.sqlserverodbc.ini` (prefixed with a period).

5. The connector searches the `/etc` directory for a file named `simba.sqlserverodbc.ini`.

Configuring ODBC Connections in Non-Windows Machine

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

- Creating a Data Source Name in Non-Windows Machine

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

- Configuring Authentication in a Non-Windows Machine

- Configuring TLS Verification in a Non-Windows Machine

- Configuring Logging Options in a Non-Windows Machine

- Testing the Connection in Non-Windows Machine

Creating a Data Source Name in 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 in a Non-Windows Machine.

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 SQL Server ODBC Connector

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

   [ODBC Data Sources]

   Sample DSN=Simba SQL Server 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/sqlserverodbc/lib/libsqlserverodbc_sbu.dylib

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

      Driver=/opt/simba/sqlserverodbc/lib/32/libsqlserverodbc_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=1500

   c. Configure authentication by specifying the authentication mechanism and your credentials. For more information, see Configuring Authentication in a Non-Windows Machine

   d. Optionally, to specify the minimum version of TLS that the connector must use to connect to the server, set the `Min_TLS` property to the required version of TLS. The supported options include `1.0` for TLS 1.0, `1.1` for TLS 1.1, and `1.2` for TLS 1.2.

   e. Optionally, encrypt your connection with TLS and configure whether the connector verifies the identity of the server. For more information, see Configuring TLS Verification in a Non-Windows Machine.

   f. 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 SQL Server ODBC Connector, see Connector Configuration Options.

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 Connector Configuration Files.

   For example, the following is an `odbc.ini` configuration file for macOS containing a DSN that connects to SQL Server using a user account:

   [ODBC Data Sources]

   Sample DSN=Simba SQL Server ODBC Connector

   [Sample DSN]

   Driver=/Library/simba/sqlserverodbc/lib/libsqlserverodbc_sbu.dylib

   Server=192.168.222.160

   Port=1500

Trusted_Connection=no

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 SQL Server using a user account:

[ODBC Data Sources]

Sample DSN=Simba SQL Server ODBC Connector 32-bit

[Sample DSN]

Driver=/opt/simba/sqlserverodbc/lib/32/libsqlserverodbc_sb32.so

Server=192.168.222.160

Port=1500

Trusted_Connection=no

UID=simba

PWD=simba123

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

Configuring a DSN-less Connection in 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 connector 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]

    Simba SQL Server ODBC Connector=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/sqlserverodbc/lib/libsqlserverodbc_sbu.dylib

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

Driver=/opt/simba/sqlserverodbc/lib/32/libsqlserverodbc_sb32.so

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

For example:

Description=Simba SQL Server 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 Connector Configuration Files.

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

[ODBC Drivers]

Simba SQL Server ODBC Connector=Installed

[Simba SQL Server ODBC Connector]

Description=Simba SQL Server ODBC Connector

Driver=/Library/simba/sqlserverodbc/lib/libsqlserverodbc_sbu.dylib

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

[ODBC Drivers]

Simba SQL Server ODBC Connector 32-bit=Installed

Simba SQL Server ODBC Connector 64-bit=Installed

[Simba SQL Server ODBC Connector 32-bit]

Description=Simba SQL Server ODBC Connector (32-bit)

Driver=/opt/simba/sqlserverodbc/lib/32/libsqlserverodbc_sb32.so

[Simba SQL Server ODBC Connector 64-bit]

Description=Simba SQL Server ODBC Connector (64-bit)

Driver=/opt/simba/sqlserverodbc/lib/64/libsqlserverodbc_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.

Configuring Authentication in a Non-Windows Machine

Connections to SQL Server require authentication. You can authenticate the connection using your SQL Server user account, the Kerberos protocol, or the NTLM protocol.

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.

> **Note:** For some of the connection properties mentioned in the instructions below, the connector accepts variant spellings of the property name. Specifically:
>
> - The connector accepts both `Integrated_Security` and `Integrated Security`.
>
> - The connector accepts both `Server_SPN` and `ServerSPN`.

Using a User Account

You can authenticate the connection by providing your user name and password for accessing the SQL Server instance.

To configure authentication using a user account:

1. Set the `Trusted_Connection` property to `false`.

2. Set the `Integrated_Security` property to `false`.

3. Set the `UID` property to an appropriate user name for accessing the server.

4. Set the `PWD` property to the password corresponding to the user name that you provided above.

Using Kerberos

You can authenticate the connection by using the Kerberos protocol.

To configure authentication using Kerberos:

1. Set the `Trusted_Connection` property to `true`.

2. Set the `Integrated_Security` property to `true`.

3. Set the `Server_SPN` property to the service principal name of the SQL Server instance.

   > **Note:** If you do not set the `Server_SPN` property, the connector uses `MSSQLSvc/[HostName]:[Port]` as the service principal name, where *[HostName]* is the IP address or host name of the server and *[Port]* is the number of the port that you are connecting to.

Using NTLM

You can authenticate the connection by using the NTLM protocol.

To configure authentication using NTLM:

1. Set the `Trusted_Connection` property to `NTLM` or to `NTLMv2` for NTLMv2.

2. Set the `Integrated_Security` property to `true`.

3. Set the `UID` property to an appropriate user name for accessing the server.

4. Set the `PWD` property to the password corresponding to the user name that you provided above.

Configuring TLS Verification in a Non-Windows Machine

If you are connecting to a SQL Server instance that has Transport Layer Security (TLS) enabled, then you can use TLS to encrypt your connection. When connecting to a server using TLS, the connector can be configured to verify the identity of the server. See Security and Authentication for more details on TLS.

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 One-way TLS Verification

You can configure the connector to verify the identity of the server.

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

1. To encrypt your connection using TLS, set the `Encrypt` property to `yes`.

2. To specify the minimum version of TLS that the connector must use to connect to the server, set the `Min_TLS` property to the required TLS version. The supported options include `1.0` for TLS 1.0, `1.1` for TLS 1.1, and `1.2` for TLS 1.2.

3. To enable one-way authentication so that the connector verifies the identity of the server, set the `TrustServerCertificate` property to `no`.

4. To specify the CA certificate that you want to use to verify the server certificate, set the `CACertificate` property to the full path and file name of the CA certificate.

Configuring a TLS Connection without Identity Verification

You can configure the connector to use TLS encryption without verifying the identity of the server.

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

1. To encrypt your connection using TLS, set the `Encrypt` property to `yes`.

2. To specify the minimum version of TLS that the connector must use to connect to the server, set the `Min_TLS` property to the required TLS version. The supported options include `1.0` for TLS 1.0, `1.1` for TLS 1.1, and `1.2` for TLS 1.2.

3. To trust the server certificate instead of verifying it, set the `TrustServerCertificate` property to `yes`.

Configuring Logging Options in a Non-Windows Machine

To help troubleshoot issues, you can enable logging in the connector or in the wire protocol component.

> **(!)** **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.sqlserverodbc.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.

Connector Logging

Use connector logging to track the activity in the Simba SQL Server ODBC Connector.

To enable logging in 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. For example:

3. Set the `LogFileCount` key to the maximum number of log files to keep.

   > **(i)** **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.

   > **(i)** **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.sqlserverodbc.ini` configuration file.

7. Restart your ODBC application to make sure that the new settings take effect.

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

- A `simbasqlserverodbcdriver.log` file that logs connector activity that is not specific to a connection.

- A `simbasqlserverodbcdriver_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.

To disable logging in a non-Windows machine:

1. Open the `simba.sqlserverodbc.ini` configuration file in a text editor.

2. Set the `LogLevel` key to `0`.

3. Save the `simba.sqlserverodbc.ini` configuration file.

4. Restart your ODBC application to make sure that the new settings take effect.

Wire Protocol Component Logging

Use wire protocol component logging to track the data activity between the connector and the SQL Server instance.

To enable wire protocol component logging on a non-Windows machine:

1. Open the `simba.sqlserverodbc.ini` configuration file in a text editor.

2. Set the `TDSTRACE` key to *[LoggingLevel]*:*[LogFilePath]*, where *[LoggingLevel]* is the logging level indicating the amount of detail to include in the log file and *[LogFilePath]* is the full path of the log file.

   The following logging levels are supported:

Logging Level	Description
0	Disables all logging.
1	Logs error events that might allow the wire protocol component to continue running. 1 is the default logging level.
2	Logs general information that describes the progress of the wire protocol component.
3	Logs detailed information that is useful for debugging the wire protocol component.
4	Logs all activity in the wire protocol component.

   For example, the following setting configures the wire protocol component to log debugging information in a file named `MyWireLog.log` located in the `C:\Logs` folder:

   TDSTRACE=3:/localhome/employee/Documents/MyWireLog.log

3. Save the `simba.sqlserverodbc.ini` configuration file.

4. Restart your ODBC application to make sure that the new settings take effect.

To disable wire protocol component logging on a non-Windows machine:

1. Set the `TDSTRACE` key to `0`.

2. Save the `simba.sqlserverodbc.ini` configuration file.

3. Restart your ODBC application to make sure that the new settings take effect.

Testing the Connection in 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.

   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 Options.

DSN Connection String Example

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

DSN=*[DataSourceName]*

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

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

DSN-less Connection String Examples

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

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

- *[PortNumber]* is the number of the TCP port that the SQL Server instance uses to listen for client connections.

- *[ServerInfo]* is the IP address or host name of the SQL Server instance to which you are connecting.

- *[ServiceName]* is the Kerberos service principal name of the SQL Server instance.

- *[SQLServerDatabase]* is the name of the database that you want to access.

- *[YourPassword]* is the password corresponding to your user name.

- *[YourPath]* is the file path corresponding to the file containing your authentication token.

- *[YourUserName]* is the user name that you use to access the SQL Server instance.

Connecting to a SQL Server Instance Using User Login

The following is an example of a DSN-less connection string for a connection to a SQL Server instance that requires user login:

Driver=Simba SQL Server ODBC Driver;Server=*[ServerInfo]*;
Port=*[PortNumber]*;Database=*[SQLServerDatabase]*;
UID=*[YourUserName]*;PWD=*[YourPassword]*;

For example:

Driver=Simba SQL Server ODBC Driver;
Server=192.168.222.160;Port=1500;Database=TestDB;
UID=simba;PWD=simba;

Connecting to a SQL Server Instance Using Kerberos Authentication

> **ⓘ Note:** For some of the connection properties included in the examples below, the connector accepts variant spellings of the property name. Specifically:
>
> - The connector accepts both `Integrated_Security` and `Integrated Security`.
>
> - The connector accepts both `Server_SPN` and `ServerSPN`.

The following is an example of a DSN-less connection string for a connection to a SQL Server instance that requires authentication through Kerberos:

Driver=Simba SQL Server ODBC Driver;Server=*[ServerInfo]*;
Port=*[PortNumber]*;Database=*[SQLServerDatabase]*;
Trusted_Connection=true;Integrated_Security=true;

Server_SPN=*[ServiceName]*;

For example:

Driver=Simba SQL Server ODBC Driver;
Server=192.168.222.160;Port=1500;Database=TestDB;
Trusted_Connection=true;Integrated_Security=true;

Server_SPN=sqls;

Connecting to a SQL Server Instance Using NTLM Authentication

> **ⓘ Note:** For the `Integrated_Security` property, which is included in the examples below, the connector also accepts `Integrated Security` as a valid spelling of the property name.

The following is an example of a DSN-less connection string for a connection to a SQL Server instance that requires authentication through the NTLM protocol:

Driver=Simba SQL Server ODBC Driver;Server=*[ServerInfo]*;
Port=*[PortNumber]*;Database=*[SQLServerDatabase]*;
Integrated_Security=true;Trusted_Connection=NTLM;
UID=*[YourUserName]*;PWD=*[YourPassword]*;

For example:

Driver=Simba SQL Server ODBC Driver;
Server=192.168.222.160;Port=1500;Database=TestDB;
Integrated_Security=true;Trusted_Connection=NTLM;

UID=domain\simba;PWD=simba;

Features

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

- Data Types

- Security and Authentication

Data Types

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

The table below lists the supported data type mappings.

SQL Server Type	SQL Type
BigInt	SQL_BIGINT
BigInt Identity	SQL_BIGINT
Binary	SQL_BINARY
Bit	SQL_BIT
Char	SQL_CHAR
Date	SQL_TYPE_DATE
DateTime	SQL_TYPE_TIMESTAMP
DateTime2	SQL_TYPE_TIMESTAMP
DateTimeOffset	SQL_SS_TIMESTAMPOFFSET
Decimal	SQL_DECIMAL
Decimal() Identity	SQL_DECIMAL
Float	SQL_FLOAT
GUID	SQL_GUID
Image	SQL_LONGVARBINARY
Int	SQL_INTEGER
Int Identity	SQL_INTEGER
Money	SQL_DECIMAL
NChar	SQL_WCHAR
NText	SQL_WLONGVARCHAR
Numeric	SQL_NUMERIC
Numeric() Identity	SQL_NUMERIC
NVarChar	SQL_WVARCHAR
NVarChar(Max)	SQL_WVARCHAR
Real	SQL_REAL
RowVersion	SQL_BINARY
SmallDateTime	SQL_TYPE_TIMESTAMP

SQL Server Type	SQL Type
SmallInt	SQL_SMALLINT
SmallInt Identity	SQL_INTEGER
SmallMoney	SQL_DECIMAL
SysName	SQL_WVARCHAR
Text	SQL_LONGVARCHAR
Time	SQL_SS_TIME2
TinyInt	SQL_TINYINT
TinyInt Identity	SQL_TINYINT
VarBinary	SQL_VARBINARY
VarBinary(Max)	SQL_VARBINARY
VarChar	SQL_VARCHAR
VarChar(Max)	SQL_VARCHAR

In addition, certain SQL Server data types are exposed via custom type identifiers, as in the native SQL Server connector:

- Geometry, Geography, and HierarchyID use custom type -151.

- XML uses custom type -152.

Security and Authentication

To protect data from unauthorized access, SQL Server data stores require connections to be authenticated with user credentials and sometimes the TLS protocol. The Simba SQL Server ODBC Connector provides full support for these authentication protocols.

The connector provides mechanisms that enable you to authenticate your connection using the Kerberos protocol, the NTLM protocol, or your SQL Server user name and password. For detailed configuration instructions, see Configuring Authentication in Windows or Configuring Authentication in a Non-Windows Machine.

Additionally, the connector supports TLS connections with or without one-way authentication. If the server has a TLS-enabled socket, then you can configure the connector to connect to it. The connector supports TLS 1.0 to 1.2. The TLS version used for the connection is the highest version that is supported by both the connector and the server.

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

Connector Configuration Options

Connector Configuration Options lists the configuration options available in the Simba SQL Server 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 Simba SQL Server ODBC Driver DSN Setup dialog box.

When using a connection string or configuring a connection from a Linux or macOS 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 SQL Server ODBC Connector, or via the key name when using a connection string or configuring a connection from a Linux or macOS machine:

▪ Application Intent	▪ Multi-Subnet Failover
▪ Application Name	▪ Password
▪ CA Certificate	▪ Port
▪ Database	▪ Return SQL Server-Specific Types as ODBC Types
▪ Description	
▪ Enable Table Types	▪ Server
▪ Encrypt	▪ Server SPN
▪ Log Level	▪ Trust Server Certificate
▪ Log Path	▪ Use NTLM
▪ Max File Size	▪ Use Trusted Connection
▪ Max Number Files	▪ User
▪ Minimum TLS/SSL	

Application Intent

This option controls whether or not the connector is in read-only mode.

- ReadWrite: The connection is not in read-only mode.

- ReadOnly: The connection is in read-only mode.

Key Name	Default Value	Required
Application_Intent	ReadWrite	No

Application Name

This option specifies the name of the application that calls SQLDriverConnect.

Key Name	Default Value	Required
Application_Name	None	No

CA Certificate

The full path and file name of the CA certificate that you want to use to verify the server certificate when TLS encryption is enabled. For information about how to trust the server certificate instead of verifying it, see Trust Server Certificate.

Key Name	Default Value	Required
CACertificate	cacerts.pem	Yes, if the Encrypt option is enabled and the Trust Server Certificate option is disabled.

Database

The name of the SQL Server database that you want to access.

Key Name	Default Value	Required
Database	None	No

Description

This description can be used to provide information about the DSN. For example, a descriptive comment about what the DSN is used for.

Key Name	Default Value	Required
Description	Simba AmazonSQL Server ODBC DSN	No

Enable Table Types

This option specifies whether the connector includes temporary tables in the results when calling SQLTables.

- Enabled (`yes`, `true`, or `1`): The connector includes temporary tables.

- Disabled (`no`, `false`, or `0`): The connector does not include temporary tables.

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

Encrypt

This option specifies whether the connector uses TLS to encrypt communication with the SQL Server instance before sending it over the network. The connector supports TLS 1.0 to 1.2. The TLS version used for the connection is the highest version that is supported by both the connector and the server.

- Enabled (`yes`, `true`, or `1`): The connector encrypts all communication with the SQL Server instance.

- Disabled (`no`, `false`, or `0`): The connector does not encrypt communication with the SQL Server instance.

For information about configuring identity verification between the connector and the server, see CA Certificate and Trust Server Certificate.

Key Name	Default Value	Required
`Encrypt`	Clear (`no`)	No

Log Level

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:

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.

Key Name	Default Value	Required
LogLevel	OFF (0)	No

Log Path

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.

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

Max File Size

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.

Key Name	Default Value	Required
LogFileSize	20971520	No

Minimum TLS/SSL

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.

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

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

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

Key Name	Default Value	Required
`Min_TLS`	When encryption is requested: 1.2 (`1.2`) When encryption is not requested: 1.0 (`1.0`)	No

Max Number Files

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.

Key Name	Default Value	Required
`LogFileCount`	50	No

Multi-Subnet Failover

This option minimizes downtime during a failover event in multi-subnet environments, improving application resilience.

Key Name	Default Value	Required
`MultiSubnetFailover`	False	No

Password

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

Key Name	Default Value	Required
`PWD`	None	Yes, if authenticating through

Key Name	Default Value	Required
		user login or the NTLM protocol.

Port

The number of the TCP port that the SQL Server instance uses to listen for client connections.

Key Name	Default Value	Required
Port	1433	Yes

Return SQL Server-Specific Types as ODBC Types

This option specifies whether the connector returns the SQL Server-specific data types that are listed in the table below as ODBC data types, or as SQL Server data types.

- Enabled (`yes`, `true`, or `1`): The connector returns the data types listed below as ODBC data types.

- Disabled (`no`, `false`, or `0`): The connector returns the data types listed below as SQL Server data types.

SQL Server Type	Returned ODBC Type
TDS_SQL_SS_TIME	SQL_TYPE_TIME
TDS_SQL_SS_ DATETIMEOFFSET	SQL_WVARCHAR

Key Name	Default Value	Required
ReturnSpecificType AsOdbcType	Clear (`no`)	No

Server

The host name or IP address of the SQL Server instance.

Key Name	Default Value	Required
Server	None	Yes

Server SPN

The service principal name of the SQL Server instance.

Key Name	Default Value	Required
Server_SPN or ServerSPN	MSSQLSvc/*[HostName]*:	No

Key Name	Default Value	Required
	[Port], where *[HostName]* is the server name or IP address specified in the Server option and *[Port]* is the port number specified in the Port option.	

Trust Server Certificate

This option specifies whether the connector trusts the server certificate when connecting to the server using TLS.

- Enabled (`yes`, `true`, or `1`): The connector trusts the server certificate.

- Disabled (`no`, `false`, or `0`): The connector does not trust the server certificate, and instead uses a CA certificate to verify the server certificate.

For information about how to specify a CA certificate, see CA Certificate.

Key Name	Default Value	Required
`TrustServerCertificate`	Clear (`no`)	No

Use NTLM

This option specifies whether the connector uses the NTLM protocol to authenticate connections to SQL Server.

- Enabled (`Integrated_Security` or `Integrated Security` is set to `true`, and `Trusted_Connection` is set to `NTLM` or to `NTLMv2`): The connector uses the NTLM protocol to authenticate connections.

- Disabled (`Integrated_Security`, `Integrated Security`, or `Trusted_Connection` is set to `false`): The connector does not use the NTLM protocol.

Names of Keys	Default Value for Each Key	Required

`Integrated_Security` or `Integrated Security`
 	
and
 	
`Trusted_Connection`
	Clear (`false`)	No

Use Trusted Connection

This option specifies whether the connector uses the Kerberos protocol to authenticate connections to SQL Server.

- Enabled (`Integrated_Security` or `Integrated Security` is set to `true`, and `Trusted_Connection` is also set to `true`): The connector uses the Kerberos protocol to

authenticate connections.

- Disabled (`Integrated_Security`, `Integrated Security`, or `Trusted_Connection` is set to `false`): The connector does not use the Kerberos protocol.

Names of Keys	Default Value for Each Key	Required

`Integrated_Security` or `Integrated Security`
 	
and
 	
`Trusted_Connection`
	Clear (`false`)	No

User

The user name that you use to access the SQL Server instance.

Key Name	Default Value	Required
`UID`	None	Yes, if authenticating through user login or the NTLM protocol.

Configuration Options Having Only Key Names

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

- Driver

- LongColumnLength

- MaxCatalogNameLen

- MaxColumnNameLen

- MaxSchemaNameLen

- MaxTableNameLen

The TDSTRACE option is available only as an environment variable.

- TDSTRACE

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

- UseLogPrefix

Driver

In Windows, the name of the installed connector for (`Simba SQL Server 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.

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

LongColumnLength

The maximum size of BLOB and CLOB columns.

Key Name	Default Value	Required
`LongColumnLength`	8000	No

MaxCatalogNameLen

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`.

Key Name	Default Value	Required
`MaxCatalogNameLen`	128	No

MaxColumnNameLen

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`.

Key Name	Default Value	Required
`MaxColumnNameLen`	128	No

MaxSchemaNameLen

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`.

Key Name	Default Value	Required
MaxSchemaNameLen	128	No

MaxTableNameLen

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`.

Key Name	Default Value	Required
MaxTableNameLen	128	No

TDSTRACE

Use this property to enable or disable logging in the wire protocol component 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.
>
> - This option is not supported in DSNs or connection strings. To configure wire protocol logging for the Windows connector, you must set TDSTRACE as an environment variable. To configure wire protocol logging for a non-Windows connector, you must set TDSTRACE in the `simba.sqlserverodbc.ini` file.

Set the property to `[LoggingLevel]:[LogFilePath]`, where *[LoggingLevel]* is the logging level indicating the amount of detail to include in the log file and *[LogFilePath]* is the fill path of the log file.

The logging levels are supported:

- `0`: Disable all logging.

- `1`: Logs error events that might allow the wire protocol component to continue running.

- `2`: Logs general information that describes the progress of the wire protocol component.

- `3`: Logs detailed information that is useful for debugging the wire protocol component.

- `4`: Logs all activity in the wire protocol component.

For example, setting `TDSTRACE` to `3:C:\Logs\MyWireLog.log` configures the wire protocol component to log debugging information in a file named `MyWireLog.log` located in the `C:\Logs` folder.

Key Name	Default Value	Required
TDSTRACE	None	No

UseLogPrefix

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.

- `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 SQL Server ODBC Connector\Driver**
    - Otherwise: **HKEY_LOCAL_MACHINE\SOFTWARE\Simba\Simba SQL Server ODBC Connector\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.sqlserverodbc.ini` file.

Key Name	Default Value	Required
UseLogPrefix	0	No

Third-Party Trademarks

Debian is a trademark or registered trademark of Software in the Public Interest, Inc. or its subsidiaries in Canada, United States and/or other countries.

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 SQL Server, SQL Server, 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.

Ubuntu is a trademark or registered trademark of Canonical Ltd. or its subsidiaries in Canada, United States and/or other countries.

All other trademarks are trademarks of their respective owners.