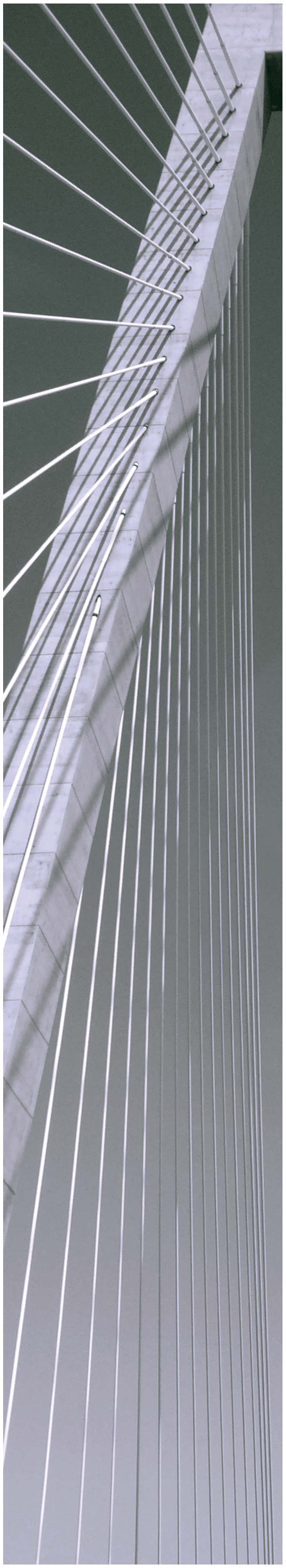

Simba Teradata ODBC Driver with SQL Connector

Installation and Configuration Guide

Simba Technologies Inc.

Version 16.20.00.1036

June 25, 2018

Contact Us

Simba Technologies Inc.
938 West 8th Avenue
Vancouver, BC Canada
V5Z 1E5

Tel: +1 (604) 633-0008

Fax: +1 (604) 633-0004

www.simba.com

About This Guide

Purpose

The *Simba Teradata ODBC Driver with SQL Connector Installation and Configuration Guide* explains how to install and configure the Simba Teradata ODBC Driver with SQL Connector. The guide also provides details related to features of the driver.

This guide supports the following releases:

- Teradata Database 16.20
- Simba Teradata ODBC Driver with SQL Connector 16.20.00.1036

This version of the Simba Teradata ODBC Driver supports the Teradata Database versions listed in the system requirements. For more information, see:

- Windows System Requirements on page 8
- macOS System Requirements on page 22
- Linux System Requirements on page 24

Audience

The guide is intended for end users of the Simba Teradata ODBC Driver, as well as administrators and developers integrating the driver.

Knowledge Prerequisites

To use the Simba Teradata ODBC Driver, the following knowledge is helpful:

- Familiarity with the platform on which you are using the Simba Teradata ODBC Driver
- Ability to use the data source to which the Simba Teradata ODBC Driver 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.

Table of Contents

About the Simba Teradata ODBC Driver

The Simba Teradata ODBC Driver enables Business Intelligence (BI), analytics, and reporting on data that is stored in Teradata Database. The driver complies with the ODBC 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 driver, 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 data residing within Teradata 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 driver in various BI tools, see the *Simba ODBC Drivers Quick Start Guide for Windows*: http://cdn.simba.com/docs/ODBC_QuickstartGuide/content/quick_start/intro.htm.

Windows Driver

Windows System Requirements

The Simba Teradata ODBC Driver supports Teradata Database versions 14.10, 15.0, 15.10, 16.0, 16.10, and 16.20.

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

- One of the following operating systems:
  - Windows 10, 8.1, or 7 SP1
  - Windows Server 2016, 2012, or 2008 R2 SP1
- 150 MB of available disk space
- Visual C++ Redistributable for Visual Studio 2012 Update 4 installed. Download and install the version that matches the bitness of the driver. You can download the installation packages at https://www.microsoft.com/en-us/download/details.aspx?id=30679.

To install the driver, you must have administrator privileges on the machine.

Installing the Driver on Windows

The Windows version of the Simba Teradata ODBC Driver is delivered in a ZIP archive named `SimbaTeradataODBC_[Version]_Windows.zip`, where *[Version]* is the version number of the driver. In addition to driver setup files and documentation, this ZIP archive contains a `SimbaTeradataODBC32_[Version].zip` archive and a `SimbaTeradataODBC64_[Version].zip` archive, each of which contain the files for the 32-bit and 64-bit drivers, respectively.

To install the Simba Teradata ODBC Driver on a Windows machine, do the following:

1. Create the installation directory by extracting the driver files from the ZIP archives and copying them to the appropriate location, depending on the bitnesses of the driver and your machine. For more information, see Creating the Installation Directory on Windows on page 9.
2. Configure the Windows Registry to recognize the driver and point to the necessary driver files. For more information, see Configuring the Windows Registry on page 10.

Make sure to install the version of the driver matching the bitness of the client application that you are using to access data in Teradata Database. For example, if you are using a 64-bit application to access Teradata Database, make sure to install the 64-bit driver.

Creating the Installation Directory on Windows

Create the installation directory for the Simba Teradata ODBC Driver on your Windows machine by extracting the driver files from the ZIP archive and copying them to the appropriate location.

On 64-bit Windows operating systems, you can execute 32- and 64-bit applications transparently. However, 64-bit applications must use 64-bit drivers, and 32-bit applications must use 32-bit drivers. Make sure that you install the version of the driver that matches the bitness of the client application.You can install both versions of the driver on the same machine.

To create the installation directory on Windows:

1. Create the driver's installation directory by doing one of the following:
   - If you are installing the 32-bit driver on a 64-bit machine, create the directory `C:\Program Files (x86)\Simba Teradata ODBC Driver\16.20`.
   - Otherwise, create the directory `C:\Program Files\Simba Teradata ODBC Driver\16.20`.

   Note:
   - If necessary, you can create an installation directory with different folder structure and naming. However, the root level folder must be named `16.20`.
   - This documentation uses the variable *[InstallDir]* to refer to this path.
2. In *`[InstallDir]`*, create a subfolder named `bin`.
3. Extract the `SimbaTeradataODBC_`*`[Version]`*`_Windows.zip` archive into a convenient temporary location.
4. Depending on whether you are installing the 32-bit or 64-bit driver, do one of the following:
   - To install the 32-bit driver, extract the `SimbaTeradataODBC32_`*`[Version]`*`.zip` archive into a convenient temporary location.
   - Or, to install the 64-bit driver, extract the `SimbaTeradataODBC64_`*`[Version]`*`.zip` archive into a convenient temporary location.
5. Copy the contents extracted from the `SimbaTeradataODBC`*`[Bitness]`*`_`*`[Version]`* folder as follows:

   a. Copy the `ErrorMessages` and `ODBC Driver for Teradata` subfolders to *`[InstallDir]`*.
   b. From the `lib` subfolder, copy all the files to *`[InstallDir]`*`\bin`.
6. From the `SimbaTeradataODBC_`*`[Version]`*`_Windows` folder, copy the `TeradataODBC.did` file to the *`[InstallDir]`*`\bin` folder.

You should now have the following subfolders in *`[InstallDir]`*:

- `bin`
- `ErrorMessages`
- `ODBC Driver for Teradata`

The `bin` subfolder should contain the following files:

- `libcrypto-1_1.dll` (for 32-bit) or `libcrypto-1_1-x64.dll` (for 64-bit)
- `sbicudt53_`*`[Bitness]`*`.dll`
- `sbicuin53_`*`[Bitness]`*`.dll`
- `sbicuuc53_`*`[Bitness]`*`.dll`
- `tdataodbc_sb`*`[Bitness]`*`.dll`
- `tdclientdir`
- `TeradataODBC.did`
- `terasso.dll`

Next, you must configure the Windows Registry. For more information, see Configuring the Windows Registry on page 10.

Configuring the Windows Registry

To complete the installation process, you need to create registry keys to do the following:

- Define the driver, specifying its location and indicating that it is installed.
- Specify driver-wide configuration settings (settings that apply to all connections that use the Simba Teradata ODBC Driver).

You can create the necessary registry keys by editing and then running the appropriate `.reg` file from the `setup` folder in the driver package.

To configure the Windows Registry:

1. In the directory where you extracted the `SimbaTeradataODBC_`*`[Version]`*`_Windows.zip` archive, browse to the `setup` folder.
2. Using a text editor, open the `.reg` file that matches the bitness of the driver and your machine:

- If you are installing the 32-bit driver on a 32-bit machine, open the `Setup-32bitDriverOn32Windows.reg` file.
- If you are installing the 32-bit driver on a 64-bit machine, open the `Setup-32bitDriverOn64Windows.reg` file.
- If you are installing the 64-bit driver on a 64-bit machine, open the `Setup-64bitDriverOn64Windows.reg` file.

3. Change all instances of `<INSTALLDIR>` to the installation directory of the driver, and then save your changes.

   Make sure to escape backslashes (`\`) by typing them twice. For example, if you installed the driver to the `C:\Program Files (x86)\Simba Teradata ODBC Driver` directory, then replace all instances of `<INSTALLDIR>` with `C:\\Program Files (x86)\\Simba Teradata ODBC Driver`.
4. Double-click the `.reg` file to run it.

The system returns a message indicating whether the registry keys were created successfully. If the keys were created successfully, you can now configure a connection and use the driver to access your Teradata data.

Creating a Data Source Name on Windows

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

Alternatively, for information about DSN-less connections, see Using a Connection String on page 41.

To create a Data Source Name on Windows:

1. Open the ODBC Data Source Administrator corresponding to the bitness of the driver that you installed.
2. 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, such as Sisense, load the data using a different user account, and might not be able to detect User DSNs that are created under another user account.

3. Click **Add**.
4. In the Create New Data Source dialog box, select **Simba Teradata ODBC Driver** and then click **Finish**. The Simba Teradata ODBC Driver DSN Setup dialog box opens.
5. In the **Name** field, type a name for your DSN.
6. Optionally, in the **Description** field, type relevant details about the DSN.
7. In the **Name or IP Address** field, type the IP address or host name of the Teradata Database instance.
8. Use the options in the Authentication area to configure authentication for your connection. For more information, see Configuring Authentication on Windows on page 13.

> **Note:**
>
> If you do not specify any authentication settings, then the driver uses the authentication mechanism specified in the `tdgssconfigure.xml` file in the TeraGSS program. Therefore, if the TeraGSS program specifies the appropriate authentication settings for your connection, you do not need to configure these settings in the driver.
>
> Typically, the TeraGSS program specifies the TD2 authentication mechanism.

9. Configure the following optional settings if needed:
   a. In the **Default Database** field, type the name of the database to access by default.
   b. In the **Account String** field, type your account string for accessing the database.
   c. To access additional optional settings, click **Options**. For more information, see Configuring Additional Driver Options on Windows on page 16.
10. From the **Session Character Set** drop-down list, select the character set to use for the session.
11. To configure logging behavior for the driver, click **Logging Options**. For more information, see Configuring Logging Options on Windows on page 19.
12. 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 Teradata ODBC Driver DSN Setup dialog box are correct. Contact your Teradata Database server administrator as needed.

13. To save your settings and close the Simba Teradata ODBC Driver DSN Setup dialog box, click **OK**.
14. To close the ODBC Data Source Administrator, click **OK**.

Configuring Authentication on Windows

Teradata databases require authentication. You can configure the Simba Teradata ODBC Driver to provide your credentials and authenticate the connection to the database using one of the following methods:

- Using Single-Sign On (SSO) on page 13
- Using TD2 on page 13
- Using LDAP on page 14
- Using Kerberos on page 14
- Using Teradata Negotiating (TDNEGO) on page 15
- Using a JSON Web Token (JWT) on page 15

Note:

If you do not specify any authentication settings, then the driver uses the authentication mechanism specified in the `tdgssconfigure.xml` file in the TeraGSS program. This is typically TD2.

Using Single-Sign On (SSO)

You can configure the driver to authenticate the connection by using Teradata Database credentials that are derived from the user information on your client machine.

To configure SSO on Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, select the DSN, and then click **Configure**.
2. From the **Mechanism** drop-down list, select the authentication mechanism that you want the driver to use.
3. Select the **Use Integrated Security** check box.
4. To save your settings and close the dialog box, click **OK**.

Using TD2

You can configure the driver to use the TD2 protocol to authenticate the connection. For this authentication mechanism, you must provide your user name and password for accessing your Teradata Database instance.

To configure TD2 authentication on Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, select the DSN, and then click **Configure**.
2. From the **Mechanism** drop-down list, select **TD2**.
3. In the **Username** field, type your Teradata Database user name.
4. Provide your password by doing one of the following:
    - Select **Password** and then type your password in the field.
    - Or, select **Teradata Wallet String** and then type your Teradata Wallet reference string in the field.

    Note:

    The Teradata Wallet utility must be installed and configured before you can connect using a reference string. For more information, see Teradata Wallet on page 48.
5. Optionally, if your database configuration requires you to specify additional parameters for authentication, click **Change**, then type the parameters in the field, and then click **OK**. For more information, see Authentication Parameter on page 55.
6. To save your settings and close the dialog box, click **OK**.

Using LDAP

You can configure the driver to use the LDAP protocol to authenticate the connection. For this authentication mechanism, you do not need to provide a user name and password. The application provides the user name and password.

To configure LDAP authentication on Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, select the DSN, and then click **Configure**.
2. From the **Mechanism** drop-down list, select **LDAP**.
3. Optionally, if your database configuration requires you to specify additional parameters for authentication, click **Change**, then type the parameters in the field, and then click **OK**. For more information, see Authentication Parameter on page 55.
4. To save your settings and close the dialog box, click **OK**.

Using Kerberos

You can configure the driver to use the Kerberos protocol to authenticate the connection. For this authentication mechanism, you do not need to provide a user name and password. The application provides the user name and password.

To configure Kerberos authentication on Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, select the DSN, and then click **Configure**.
2. From the **Mechanism** drop-down list, select **KRB5**.
3. Optionally, if your database configuration requires you to specify additional parameters for authentication, click **Change**, then type the parameters in the field, and then click **OK**. For more information, see Authentication Parameter on page 55.
4. To save your settings and close the dialog box, click **OK**.

Using Teradata Negotiating (TDNEGO)

You can configure the driver to select the authentication mechanism to use through Teradata Negotiating. Depending on the mechanism that the driver selects as a result of the negotiation process, you might need to provide a user name and password.

To configure TDNEGO authentication on Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, select the DSN, and then click **Configure**.
2. From the **Mechanism** drop-down list, select **TDNEGO**.
3. Optionally, if your database configuration requires you to specify additional parameters for authentication, click **Change**, then type the parameters in the field, and then click **OK**. For more information, see Authentication Parameter on page 55.
4. To save your settings and close the dialog box, click **OK**.

Using a JSON Web Token (JWT)

You can configure the driver to authenticate the connection using a token obtained from the UDA User Service.

To configure JWT authentication on Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, select the DSN, and then click **Configure**.
2. From the **Mechanism** drop-down list, select **JWT**.
3. In the **Parameter** field, type the following, where *[JWT_Token]* is the JSON web token that you obtained from the UDA User Service:

   ```
   token=[JWT_Token]
   ```

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

Configuring Additional Driver Options on Windows

You can configure additional options to modify the behavior of the driver.

To configure additional driver options on Windows:

1. To access additional options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Options**.
2. To return column names instead of column titles when retrieving data, select the **Use Column Names** check box.
3. To use X views so that the driver can only access objects that the specified user owns or controls, select the **Use X View** check box.
4. To disable the HELP database, select the **No HELP DATABASE** check box.
5. To treat the underscore (_) and percent sign (%) characters as normal characters instead of search wildcards, select the **Ignore Search Patterns** check box.
6. To disable the driver SQL parser and pass SQL statements through to the database unchanged, select the **Disable Parsing** check box.
7. To log error events in the Event Viewer of the Teradata server, select the **Log Error Events** check box.
8. To display decimal symbols based on regional settings, select the **Use Regional Settings for Decimal Symbol** check box.
9. To encrypt all communication between the driver and the database, select the **Enable Data Encryption** check box.
10. To use extended statement information and enable support for the SQLDescribeParam ODBC API function, select the **Enable Extended Statement Information** check box.
11. To specify the session mode that the driver uses during sessions on the database, from the **Session Mode** drop-down list, select the appropriate mode.
12. To specify the format that the driver uses for DATE values when communicating with the database, from the **Date Time Format** drop-down list, select **AAA** for ANSI format or **IAA** for Integer format.
13. To specify how auto-generated keys are returned for requests that insert data into identity columns, from the **Return Generated Keys** drop-down list, select one of the following methods:
    - **Whole Row**: The entire row is returned.
    - **Identity Column**: Only data from the identity column is returned.
    - **No**: Auto-generated keys are not returned.
14. To specify whether the driver supports Unicode Pass Through (UPT) for Pass Through Characters (PTCs), from the **UPT Mode** drop-down list, select one of the following settings:

- **Notset**: The driver does not do anything to change UPT support.
- **UPTON**: The driver sends a query to the database to enable UPT support.
- **UPTOFF**: The driver sends a query to the database to disable UPT support.

Note:

For more information about UPT, see "Unicode Pass Through" in the Teradata Database documentation: http://info.teradata.com/htmlpubs/DB_TTU_16_00/index.html#page/General_Reference/B035-1098-160K/ifk1472240714022.html.

15. To configure advanced driver options, click **Advanced**. For more information, see Configuring Advanced Options on Windows on page 17.

! Important:

Do not modify the advanced driver options unless your system administrator instructs you to do so. These options are needed in specific scenarios only, and may cause unexpected driver behavior if not configured appropriately.

16. To save your settings and close the Simba Teradata ODBC Driver Options dialog box, click **OK**.

Configuring Advanced Options on Windows

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

! Important:

Do not modify the advanced driver options unless your system administrator instructs you to do so. These options are needed in specific scenarios only, and may cause unexpected driver behavior if not configured appropriately.

To configure advanced options on Windows:

1. To access advanced options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, then click **Options**, then click **Advanced**.
2. In the **Maximum Response Buffer** field, specify the maximum size of the response buffer for SQL requests, in kilobytes.
3. In the **TDMST Port Number** field, specify the number of the port used to access Teradata Database.
4. In the **Translation DLL Name** field, specify the `.dll` file that contains functions for translating all the data that is transferred between the Teradata server and the

driver. This `.dll` file is used for translation if local character sets are not supported by Teradata Database or the driver.

5. In the **Translation Option** field, specify the options used by the Translation DLL file. The required options may vary depending on the Translation DLL file being used.
6. In the **Login Timeout** field, specify the number of seconds to wait for a response when logging in to the database.
7. To enable the print option when creating stored procedures, from the **Procedure With Print Stmt** drop-down list, select **P**.
8. To enable the SPL option when creating stored procedures, from the **Procedure With SPL Source** drop-down list, select **Y**.
9. In the **Data Source DNS Entries** field, specify how the driver determines which DNS entry to use by doing one of the following:
   - To resolve DNS entries dynamically, leave the field empty.
   - Or, to use DNS lookup, type **0**.
   - Or, to specify a number of DNS entries to use in a round-robin fashion, type the number of entries.
10. To use the TCP_NODELAY setting, select the **Use TCP_NODELAY** check box.
11. To specify NULL for the Catalog Name parameter in all Catalog API functions, select the **Use NULL For Catalog Name** check box.
12. To have the driver request the next response message while it is processing the current response message, select the **Enable Read Ahead** check box.
13. To retry socket system calls at the driver level instead of the application level, select the **Retry System Calls (EINTR)** check box.
14. To optimize retrieval for Large Object (LOB) data that meets specified size requirements, enable Smart LOB (SLOB) Mode by doing the following. For detailed information about the supported LOB retrieval modes, see LOB Retrieval Modes on page 49.
    a. In the **Max Single LOB Bytes** field, type the maximum size of the LOBs (in bytes) that the driver can retrieve using SLOB Mode. LOBs that exceed this size are retrieved using Deferred Mode instead.
    b. In the **Max Total LOB Bytes Per Row** field, type the maximum size of LOB data per row (in bytes) that the driver can retrieve using SLOB Mode. If the total amount of LOB data being retrieved from a row exceeds this size, then after using SLOB Mode to retrieve LOBs up to this size limit, the driver uses Deferred Mode to retrieve the remaining LOBs from that row.
    c. If you are retrieving LOB data from columns in sequential order, select the **Use Sequential Retrieval Only** check box.

> ! **Important:**
>
> If you enable this option but then retrieve LOB data from columns in a non-sequential order, driver performance may decrease. In this scenario, the driver discards the LOBs that are returned through SLOB Mode and must then retrieve them all again using Deferred Mode.

15. To enable compatibility with applications that use Microsoft Access Jet databases by using DATE data in TIMESTAMP parameters, select the **Use DATE Data For TIMESTAMP Parameters** check box.
16. To provide backwards compatibility for ODBC 2.x applications that use noncompliant search patterns, select the **Enable Custom Catalog Mode For 2.x Applications** check box.
17. To return an empty string in the CREATE_PARAMS column when you call SQLGetTypeInfo for SQL_TIMESTAMP data, select the **Return Empty String In CREATE_PARAMS Column For SQL_TIMESTAMP** check box.
18. To return a hard-coded value as the maximum length of SQL_CHAR and SQL_VARCHAR columns, select the **Return Max CHAR/VARCHAR Length As 32K** check box.

> **Note:**
>
> - Enabling this option prevents the returned column size from causing numeric overflows in Microsoft Access.
> - The hard-coded value is either 32000 or 64000, depending on the setting specified for the Session Character Set driver option.

19. To save your settings and close the Advanced Options 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 Teradata ODBC Driver, 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 Teradata ODBC Driver, so make sure to disable the feature after you are done using it.

To enable driver 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 driver to abort.
ERROR	Logs error events that might allow the driver 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 driver.
DEBUG	Logs detailed information that is useful for debugging the driver.
TRACE	Logs all driver 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 driver 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 driver 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 Teradata ODBC Driver produces two log files at the location you specify in the Log Path field, where *[DriverName]* is the name of the driver:

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

To disable driver 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.

Verifying the Driver Version Number on Windows

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

To verify the driver version number on Windows:

1. Open the ODBC Data Source Administrator corresponding to the bitness of the driver that you installed.
2. Click the **Drivers** tab and then find the Simba Teradata ODBC Driver in the list of ODBC drivers that are installed on your system. The version number is displayed in the **Version** column.

macOS Driver

macOS System Requirements

The Simba Teradata ODBC Driver supports Teradata Database versions 14.10, 15.0, 15.10, 16.0, 16.10, and 16.20.

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

- macOS version 10.11, 10.12, or 10.13
- 100 MB of available disk space
- iODBC 3.52.9, 3.52.10, 3.52.11, or 3.52.12

Installing the Driver on macOS

The macOS version of the Simba Teradata ODBC Driver is delivered in a tarball named `SimbaTeradataODBC_[Version]-OSX.tar.gz`, where *[Version]* is the version number of the driver.

To install the Simba Teradata ODBC Driver on a macOS machine, create the installation directory by extracting the driver files from the tarball and copying them to the appropriate locations. Then, configure the environment variables on your machine to make sure that the ODBC driver manager can work with the driver.

To install the driver on macOS:

1. Create the following directory: `/Library/Application Support/Simba Teradata ODBC Driver/16.20`

   Note:
   - If necessary, you can create an installation directory with different folder structure and naming. However, the root level folder must be named `16.20`.
   - This documentation uses the variable *[InstallDir]* to refer to this path.
2. Extract the `SimbaTeradataODBC-[Version]-OSX.tar.gz` file into a convenient temporary location.
3. From the `SimbaTeradataODBC-[Version]-OSX` subfolder, extract the `SimbaTeradataODBC-[Version].tar.gz` file.
4. From the `SimbaTeradataODBC-[Version]` folder, copy the following files and folders to `[InstallDir]`:

   - The `ErrorMessages` subfolder.
   - The `lib` subfolder.
5. From the `SimbaTeradataODBC-`*`[Version]`*`-OSX` folder, copy the following files and folders to the *`[InstallDir]`*`/lib` directory:
   - The `TeradataODBC.did` file.
   - The `odbc.ini`, `odbcinst.ini`, and `simba.teradataodbc.ini` files from the `setup` subfolder.

You should now have the following file and subfolders in *`[InstallDir]`*:

- `ErrorMessages`
- `lib`

The `lib` subfolder should contain the following files:

- `libtdsso.dylib`
- `tdataodbc_sbu.dylib`
- `tdclientdir`
- `TeradataODBC.did`
- `odbc.ini`
- `odbcinst.ini`
- `simba.teradataodbc.ini`

Next, configure the environment variables on your machine to make sure that the ODBC driver manager can work with the driver. For more information, see Configuring the ODBC Driver Manager on Non-Windows Machines on page 27.

Linux Driver

Linux System Requirements

The Simba Teradata ODBC Driver supports Teradata Database versions 14.10, 15.0, 15.10, 16.0, 16.10, and 16.20.

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

- One of the following distributions:
  - Red Hat® Enterprise Linux® (RHEL) 6 or 7
  - CentOS 6 or 7
  - SUSE Linux Enterprise Server (SLES) 11 or 12
  - Debian 8 or 9
  - Ubuntu 16.04
- 150 MB of available disk space
- One of the following ODBC driver managers installed:
  - iODBC 3.52.9, 3.52.10, 3.52.11, or 3.52.12
  - unixODBC 2.3.2, 2.3.3, or 2.3.4

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

Installing the Driver on Linux

The Linux version of the Simba Teradata ODBC Driver is delivered through a tarball file named `TeradataODBC_[Version]-Linux.tar.gz`, where *[Version]* is the version number of the driver.

To install the Simba Teradata ODBC Driver on a Linux machine, create the installation directory by extracting the files from the tarball and copying them to the appropriate locations. Then, configure the environment variables on your machine to make sure that the ODBC driver manager can work with the driver.

To install the driver on Linux:

1. Create the following directory: `/opt/teradata/client/16.20/`.

   Note:
   - If necessary, you can create an installation directory with different folder structure and naming. However, the root level folder must be named `16.20`.
   - This documentation uses the variable *[InstallDir]* to refer to this path.

2. In *`[InstallDir]`*, create one of the following subfolders:
   - If you are installing the 32-bit driver, create a subfolder named `lib`.
   - Or, if you are installing the 64-bit driver, create a subfolder named `lib64`.
3. Extract the `TeradataODBC_`*`[Version]`*`-Linux.tar.gz` file into a convenient temporary location.
4. From the `TeradataODBC_`*`[Version]`*`-Linux` folder, copy the following files into the *`[InstallDir]`*`/lib` directory (for the 32-bit driver) or the *`[InstallDir]`*`/lib64` directory (for the 64-bit driver):
   - `TeradataODBC.did`
   - The files inside the `setup` folder (`odbc.ini`, `odbcinst.ini`, and `simba.teradataodbcodbc.ini`).
5. Depending on whether you are installing the 32-bit or 64-bit driver, do one of the following:
   - To install the 32-bit driver, extract the `SimbaTeradataODBC32_`*`[Version]`*`.tar.gz` file into a convenient temporary location.
   - To install the 64-bit driver, extract the `SimbaTeradataODBC64_`*`[Version]`*`.tar.gz` file into a convenient temporary location.
6. Copy the contents of the `SimbaTeradataODBC`*`[Bitness]`*`_`*`[Version]`* folder as follows:
   a. Copy the `ErrorMessages` folder to *`[InstallDir]`*.
   b. Copy all the files from the `lib` subfolder to *`[InstallDir]`*`/lib` (for the 32-bit driver) or *`[InstallDir]`*`/lib64` (for the 64-bit driver).

You should now have the following file and folder structure in the *`[InstallDir]`*:

- `/ErrorMessages`
- `/lib` (for the 32-bit driver) or `lib64` (for the 64-bit driver)
  - `libtdsso.so`
  - `odbc.ini`
  - `odbcinst.ini`
  - `simba.teradataodbc.ini`
  - `tdataodbc_sb`*`[Bitness]`*`.so`

  - `tdclientdir`
  - `TeradataODBC.did`

Next, configure the environment variables on your machine to make sure that the ODBC driver manager can work with the driver. For more information, see Configuring the ODBC Driver Manager on Non-Windows Machines on page 27.

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 Teradata ODBC Driver, 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 27.
- If the driver 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 28.

After configuring the ODBC driver manager, you can configure a connection and access your data store through the driver. For more information, see Configuring ODBC Connections on a Non-Windows Machine on page 30.

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 driver. 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.

Troubleshooting

When you attempt to connect through the driver on a Linux machine, you may encounter the following error message:

```
SQLDriverConnect = [Simba][ODBC] (11560) Unable to locate
SQLGetPrivateProfileString function. (11560)
```

This issue may occur when the name of the library file for the driver manager is different from the default. To resolve this issue, do the following:

1. Confirm the name of the library file that is used by your driver manager.
2. In a text editor, open the `simba.teradataodbc.ini` file (located in `[InstallDir]/lib` by default).
3. Add the following line to the end of the file, where *[DMLibFile]* is the name of the library file:

   ```
   ODBCInstLib=[DMLibFile]
   ```

4. Save the `simba.teradataodbc.ini` file.

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.teradataodbc.ini` file in the `lib` subfolder of the driver 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 SIMBAODBCINI to the full path and file name of the `simba.teradataodbc.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 SIMBAODBCINI to the full path and file name of the `simba.teradataodbc.ini` file.

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

For unixODBC:

```
export ODBCINI=/usr/local/odbc/odbc.ini
export ODBCSYSINI=/usr/local/odbc
export SIMBAODBCINI=/etc/simba.teradataodbc.ini
```

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

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

Configuring ODBC Connections on a Non-Windows Machine

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

- Creating a Data Source Name on a Non-Windows Machine on page 30
- Configuring a DSN-less Connection on a Non-Windows Machine on page 32
- Configuring Authentication on a Non-Windows Machine on page 34
- Configuring Logging Options on a Non-Windows Machine on page 37
- Testing the Connection on a Non-Windows Machine on page 38

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.

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

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

   For example, on a macOS machine:

   ```
   [ODBC Data Sources]
   Sample DSN=Teradata ODBC Driver
   ```

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

   ```
   [ODBC Data Sources]
   Sample DSN=Teradata ODBC Driver 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 driver library file that matches the bitness of the application.

      For example, on a macOS machine:

```
Driver=/Library/Application Support/Simba Teradata
ODBC Driver/16.20/lib/tdataodbc_sbu.dylib
```

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

```
Driver=/opt/teradata/client/16.20/lib/tdataodbc_
sb32.so
```

b. Set the `DBCName` property to the IP address or host name of the Teradata Database instance.

   For example:

```
DBCName=192.168.222.160
```

c. Configure authentication for your connection by specifying the authentication mechanism and your credentials as needed. For more information, see Configuring Authentication on a Non-Windows Machine on page 34.

Note:

If you do not specify any authentication settings, then the driver uses the authentication mechanism that the TeraGSS program specifies in the `tdgssconfigure.xml` file. Therefore, if the TeraGSS program specifies the appropriate authentication settings for your connection, you do not need to configure these settings in the driver.

Typically, the TeraGSS program specifies TD2 as the authentication mechanism to use.

d. Optionally, set additional key-value pairs as needed to specify other connection settings. For detailed information about each connection property, see Configuration Options Appearing in the User Interface on page 53.

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

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

```
[ODBC Data Sources]
Sample DSN=Teradata ODBC Driver
[Sample DSN]
Driver=/Library/Application Support/Simba Teradata ODBC
Driver/16.20/lib/tdataodbc_sbu.dylib
DBCName=192.168.222.160
MechanismName=TD2
```

```
UID=jsmith
PWD=simba123
```

As another example, the following is an `odbc.ini` configuration file for a 32-bit driver on a Linux machine, containing a DSN that connects to Teradata:

```
[ODBC Data Sources]
Sample DSN=Teradata ODBC Driver 32-bit
[Sample DSN]
Driver=/opt/teradata/client/16.20/lib/tdataodbc_sb32.so
DBCName=192.168.222.160
MechanismName=TD2
UID=jsmith
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 driver in the `odbcinst.ini` file and then provide a DSN-less connection string in your application.

To define a driver on a non-Windows machine:

1. In a text editor, open the `odbcinst.ini` configuration file.
2. In the `[ODBC Drivers]` section, add a new entry by typing a name for the driver, an equal sign (=), and then `Installed`.

   For example, on a macOS machine:

   ```
   [ODBC Drivers]
   Teradata ODBC Driver=Installed
   ```

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

   ```
   [ODBC Drivers]
   Teradata ODBC Driver 32-bit=Installed
   ```

3. Create a section that has the same name as the driver (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 driver library file that matches the bitness of the application.

   For example, on a macOS machine:

   ```
   Driver=/Library/Application Support/Simba Teradata
   ODBC Driver/16.20/lib/tdataodbc_sbu.dylib
   ```

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

   ```
   Driver=/opt/teradata/client/16.20/lib/tdataodbc_
   sb32.so
   ```

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

   For example:

   ```
   Description=Teradata ODBC Driver
   ```

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

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

```
[ODBC Drivers]
Teradata ODBC Driver=Installed
[Teradata ODBC Driver]
Driver=/Library/Application Support/Simba Teradata ODBC
Driver/16.20/lib/tdataodbc_sbu.dylib
Description=Teradata ODBC Driver
```

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

```
[ODBC Drivers]
Teradata ODBC Driver 32-bit=Installed
Teradata ODBC Driver 64-bit=Installed
[Teradata ODBC Driver 32-bit]
Driver=/opt/teradata/client/16.20/lib/tdataodbc_sb32.so
Description=Teradata ODBC Driver (32-bit)
[Teradata ODBC Driver 64-bit]
Driver=/opt/teradata/client/16.20/lib64/tdataodbc_sb64.so
Description=Teradata ODBC Driver (64-bit)
```

You can now connect to your data store by providing your application with a connection string where the `Driver` property is set to the driver 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 41.

For detailed information about all the connection properties that the driver supports, see Driver Configuration Options on page 53.

Configuring Authentication on a Non-Windows Machine

Teradata databases require authentication. You can configure the Simba Teradata ODBC Driver to provide your credentials and authenticate the connection to the database using one of the following methods:

- Using Single-Sign On (SSO) on page 34
- Using TD2 on page 34
- Using LDAP on page 35
- Using Kerberos on page 35
- Using Teradata Negotiating (TDNEGO) on page 36
- Using a JSON Web Token (JWT) on page 36

Note:

If you do not specify any authentication settings, then the driver uses the authentication mechanism specified in the `tdgssconfigure.xml` file in the TeraGSS program. This is typically TD2.

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.

Using Single-Sign On (SSO)

You can configure the driver to authenticate the connection by using Teradata Database credentials that are derived from the user information on your client machine.

To configure SSO on a non-Windows machine:

1. Set the `MechanismName` property to `TD2`.
2. Set the `UseIntegratedSecurity` property to `1`.

Using TD2

You can configure the driver to use the TD2 protocol to authenticate the connection. For this authentication mechanism, you must provide your user name and password for accessing your Teradata Database instance.

To configure TD2 authentication on a non-Windows machine:

1. Set the `MechanismName` property to `TD2`.
2. Set the `UID` property to your Teradata Database user name..
3. Set the `Password` property to one of the following:
   - Your Teradata Database password.
   - Or, your Teradata Wallet reference string, using the following format where *[WalletString]* is your reference string:

   ```
   $tdwallet([WalletString])
   ```

   Note:

   The Teradata Wallet utility must be installed and configured before you can connect using a reference string. For more information, see Teradata Wallet on page 48.
4. Optionally, if your database configuration requires you to specify additional parameters for authentication, set the `AuthenticationParameter` property to those parameters. For more information, see Authentication Parameter on page 55.

Using LDAP

You can configure the driver to use the LDAP protocol to authenticate the connection. For this authentication mechanism, you do not need to provide a user name and password. The application provides the user name and password.

To configure LDAP authentication on a non-Windows machine:

1. Set the `MechanismName` property to `LDAP`.
2. Optionally, if your database configuration requires you to specify additional parameters for authentication, set the `AuthenticationParameter` property to those parameters. For more information, see Authentication Parameter on page 55.

Using Kerberos

You can configure the driver to use the Kerberos protocol to authenticate the connection. For this authentication mechanism, you do not need to provide a user name and password. The application provides the user name and password.

To configure Kerberos authentication on a non-Windows machine:

1. Set the `MechanismName` property to `KRB5`.
2. Optionally, if your database configuration requires you to specify additional parameters for authentication, set the `AuthenticationParameter` property to those parameters. For more information, see Authentication Parameter on page 55.

Using Teradata Negotiating (TDNEGO)

You can configure the driver to select the authentication mechanism to use through Teradata Negotiating. Depending on the mechanism that the driver selects as a result of the negotiation process, you might need to provide a user name and password.

To configure TDNEGO authentication on a non-Windows machine:

1. Set the `MechanismName` property to `TDNEGO`.
2. Optionally, if your database configuration requires you to specify additional parameters for authentication, set the `AuthenticationParameter` property to those parameters. For more information, see Authentication Parameter on page 55.

Using a JSON Web Token (JWT)

You can configure the driver to authenticate the connection using a token obtained from the UDA User Service.

To configure JWT authentication on a non-Windows machine:

1. Set the `MechanismName` property to `JWT`.
2. Set the `AuthenticationParameter` property to the following, where *[JWT_Token]* is the JSON web token that you obtained from the UDA User Service:

   ```
   token=[JWT_Token]
   ```

   For example, if your token is zio5YOBZ.nExFB6lm.SOwvlWy2, then you set the `AuthenticationParameter` as follows:

   ```
   AuthenticationParameter=
   {token=zio5YOBZ.nExFB6lm.SOwvlWy2}
   ```

   For more information about this property, see Authentication Parameter on page 55.

Configuring Logging Options on a Non-Windows Machine

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

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

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

To enable logging on a non-Windows machine:

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

LogLevel Value	Description
`0`	Disables all logging.
`1`	Logs severe error events that lead the driver to abort.
`2`	Logs error events that might allow the driver 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 driver.
`5`	Logs detailed information that is useful for debugging the driver.
`6`	Logs all driver activity.

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

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

5. Set the `LogFileSize` key to the maximum size of each log file in megabytes (MB).

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

6. Save the `simba.teradataodbc.ini` configuration file.
7. Restart your ODBC application to make sure that the new settings take effect.

The Simba Teradata ODBC Driver produces two log files at the location you specify using the `LogPath` key, where *[DriverName]* is the name of the driver:

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

To disable logging on a non-Windows machine:

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

Testing the Connection on a Non-Windows Machine

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

Using the iODBC Driver Manager

You can use the iodbctest and iodbctestw utilities to establish a test connection with your driver. Use iodbctest to test how your driver works with an ANSI application, or use iodbctestw to test how your driver 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 41.

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 driver and your DSN. isql and iusql can only be used to test connections that use a DSN. Use isql to test how your driver works with an ANSI application, or use iusql to test how your driver 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 Driver Configuration Options on page 53.

DSN Connection String Example

To write a connection string that uses a DSN, set the `DSN` key to the name of your DSN. As an alternative, you can set the `DataSourceName` key, which is synonymous with the `DSN` key.

The following are examples of connection strings that use a DSN:

```
DSN=MyDSNForTeradata
```

```
DataSourceName=MyDSNForTeradata
```

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 driver without a DSN. To connect to a data source without using a DSN, use a connection string instead.

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

- *[AuthenticationMechanism]* is the mechanism that the driver uses to authenticate the connection to the database. For information about the supported settings, see Mechanism on page 64.
- *[JWT_Token]* is the JSON web token that you obtained from the UDA User Service.

- *[Server]* is the IP address or host name of the Teradata Database instance to which you are connecting.
- *[YourUserName]* is the user name that you use to access the database.
- *[YourPassword]* is the password corresponding to your user name.

Connecting to a Teradata Database Instance Using Single Sign-On

The following is the format of a DSN-less connection string that connects to the database using Single Sign-On (SSO):

```
Driver=SimbaTeradata ODBC Driver;DBCName=
[Server];MechanismName=
[AuthenticationMechanism];UseIntegratedSecurity=1;
```

Note:

`MechanismName` is optional. If this option is not set, then the driver uses the authentication mechanism that the TeraGSS program specifies in the `tdgssconfigure.xml` file.

For example:

```
Driver=SimbaTeradata
ODBC Driver;DBCName=192.168.222.160;MechanismName=TD2;UseInt
egratedSecurity=1;
```

Connecting to a Teradata Database Instance Using TD2

The following is the format of a DSN-less connection string that connects to the database using the TD2 protocol:

```
Driver=Simba Teradata ODBC Driver;DBCName=
[Server];MechanismName=TD2;UID=[YourUserName];PWD=
[YourPassword];
```

For example:

```
Driver=Simba Teradata ODBC
Driver;DBCName=192.168.222.160;MechanismName=TD2;
UID=jsmith;PWD=simba123;
```

Alternatively, you can provide a Teradata Wallet reference string instead of a password. For example:

```
Driver=Simba Teradata ODBC
Driver;DBCName=192.168.222.160;MechanismName=TD2;
UID=jsmith;PWD=$tdwallet(jsmith_wallet_
string);EnableWallet=1;
```

Note:

The Teradata Wallet utility must be installed and configured before you can connect using a reference string. For more information, see Teradata Wallet on page 48.

Connecting to a Teradata Database Instance Using LDAP, Kerberos, or TDNEGO

The following is the format of a DSN-less connection string that connects to the database using the LDAP, Kerberos, or TDNEGO protocol. For LDAP and Kerberos, you do not need to specify the set `UID` and `PWD` properties, because the driver obtains these credentials from the application. For TDNEGO, depending on the actual mechanism that the driver selects as a result of the negotiation process, you might need to set `UID` and `PWD` as shown in the example above for TD2.

```
Driver=Simba Teradata ODBC Driver;DBCName=
[Server];MechanismName=[AuthenticationMechanism];
```

For example, to use LDAP:

```
Driver=SimbaTeradata ODBC
Driver;DBCName=192.168.222.160;MechanismName=LDAP;
```

Connecting to a Teradata Database Instance Using a JSON Web Token

The following is the format of a DSN-less connection string that connects to the database using a JSON web token (JWT):

```
Driver=Simba Teradata ODBC Driver;DBCName=
[Server];MechanismName=JWT;AuthenticationParameter={token=
[JWT_Token]};
```

For example:

```
Driver=SimbaTeradata ODBC
Driver;DBCName=192.168.222.160;MechanismName=JWT;
AuthenticationParameter={token=zio5YOBZ.nExFB6lm.SOwvlWy2};
```

Features

For more information on the features of the Simba Teradata ODBC Driver, see the following:

- Data Types on page 44
- Authentication and Encryption on page 48
- Teradata Wallet on page 48
- LOB Retrieval Modes on page 49
- Scalar Function Support on page 51
- Special Query Syntax on page 51

Data Types

The Simba Teradata ODBC Driver supports two-way mapping between Teradata SQL types and many common ODBC SQL data types.

The tables below list the supported data types and their mappings. The first table lists Teradata SQL types that are mapped to standard ODBC SQL data types, while the second table lists those that are mapped to custom SQL types.

Note:

As indicated below, some Teradata SQL types may return differently depending on the character set that is specified in the Session Character Set option (the `CharacterSet` key). For more information, see Session Character Set on page 68.

Teradata SQL Type	ODBC SQL Type
BIGINT	SQL_BIGINT
BLOB	SQL_LONGVARBINARY
BYTE	SQL_BINARY
BYTEINT	SQL_TINYINT

Teradata SQL Type	ODBC SQL Type
CHARACTER	SQL_CHAR when using a non-Unicode character set. SQL_WCHAR when using a Unicode character set.
CLOB	SQL_LONGVARCHAR when using a non-Unicode character set. SQL_WLONGVARCHAR when using a Unicode character set.
DATE	SQL_TYPE_DATE
DECIMAL_1 DECIMAL_2 DECIMAL_4 DECIMAL_8 DECIMAL_16 FIXED_NUMBER	SQL_NUMERIC SQL_DECIMAL
FLOAT FLOATING_NUMBER	SQL_DOUBLE
INTEGER	SQL_INTEGER
INTERVAL_DAY	SQL_INTERVAL_DAY
INTERVAL_HOUR	SQL_INTERVAL_HOUR
INTERVAL_MINUTE	SQL_INTERVAL_MINUTE
INTERVAL_MONTH	SQL_INTERVAL_MONTH
INTERVAL_SECOND	SQL_INTERVAL_SECOND

Teradata SQL Type	ODBC SQL Type
INTERVAL_YEAR	SQL_INTERVAL_YEAR
INTERVAL_DAY_TO_HOUR	SQL_INTERVAL_DAY_TO_HOUR
INTERVAL_DAY_TO_ MINUTE	SQL_INTERVAL_DAY_TO_MINUTE
INTERVAL_DAY_TO_ SECOND	SQL_INTERVAL_DAY_TO_SECOND
INTERVAL_HOUR_TO_ MINUTE	SQL_INTERVAL_HOUR_TO_MINUTE
INTERVAL_HOUR_TO_ SECOND	SQL_INTERVAL_HOUR_TO_SECOND
INTERVAL_MINUTE_TO_ SECOND	SQL_INTERVAL_MINUTE_TO_SECOND
INTERVAL_YEAR_TO_ MONTH	SQL_INTERVAL_YEAR_TO_MONTH
SMALLINT	SQL_SMALLINT
TIME TIME_WTZ	SQL_TYPE_TIME
TIMESTAMP TIMESTAMP_WTZ	SQL_TYPE_TIMESTAMP
VARCHAR	SQL_VARCHAR when using a non-Unicode character set. SQL_WVARCHAR when using a Unicode character set.
VARBYTE	SQL_VARBINARY

Teradata SQL Type	ODBC SQL Type
FLOAT	SQL_FLOAT SQL_REAL

The following table lists Teradata SQL types that are mapped to custom SQL types.

Teradata SQL Type	Custom SQL Types
AVRO	SQL_TD_DATASET_AVRO (18006)
CSV	SQL_TD_DATASET_CSV (18007) when using a non-Unicode character set. SQL_TD_DATASET_WCSV (18008) when using a Unicode character set.
JSON	SQL_TD_JSON (18004) when using a non-Unicode character set. SQL_TD_WJSON (18005) when using a Unicode character set.
PERIOD_DATE	SQL_PERIOD_DATE (-1049)
PERIOD_TIME	SQL_PERIOD_TIME (-1048)
PERIOD_TIME_TZ	SQL_PERIOD_TIME_WITH_TIME_ZONE (-1047)
PERIOD_TIMESTAMP	SQL_PERIOD_TIMESTAMP (-1046)
PERIOD_TIMESTAMP_TZ	SQL_PERIOD_TIMESTAMP_WITH_TIME_ZONE (-1045)
FIXED_NUMBER	SQL_TD_FIXED_NUMBER (18001) == SQL_DECIMAL
FLOATING_NUMBER	SQL_TD_FLOATING_NUMBER (18002) == SQL_DOUBLE
XML	SQL_TD_XML (18003)

Authentication and Encryption

Teradata Database secures data by requiring authentication for access. To access your data, you must configure the driver to pass in your credentials and authenticate the connection. The Simba Teradata ODBC Driver supports a number of methods for authenticating connections:

- TD2
- Kerberos (KRB5)
- LDAP
- JSON Web Token (JWT)
- Teradata Negotiating (TDNEGO)
- Single Sign-On (SSO), including SSO through TDNEGO.

Configure authentication for your connection by selecting an authentication mechanism and then specifying the appropriate credentials in the DSN or connection string, if needed. When you use LDAP or KRB5 (Kerberos), the driver uses the credentials from the application. If Teradata Wallet has been configured for your credentials, you can specify your Teradata Wallet reference string instead of your password. For detailed configuration instructions, see Configuring Authentication on Windows on page 13 or Configuring Authentication on a Non-Windows Machine on page 34.

In addition to authentication for database access, the driver also supports encryption for any data that is passed between the driver and the database. You can configure the Enable Data Encryption option (the `UseDataEncryption` key) to specify whether the driver encrypts all communication with the database or authentication information only.

Teradata Wallet

Teradata Wallet is a software package that secures Teradata Database passwords on client machines. It maps your password to a reference string, which you can use instead of your password during authentication. Providing your reference string instead of your password lets you obscure your password.

Teradata Wallet is installed and configured separately from the driver. To download the software package, go to http://downloads.teradata.com and click the Teradata Wallet link for the platform that you are using. For information about configuring Teradata Wallet, see "Introducting Teradata Wallet" on the Teradata Developer Exchange: http://developer.teradata.com/tools/articles/introducing-teradata-wallet.

After Teradata Wallet is set up, you connect to the database using your reference string instead of your password. When specifying your connection information through the Simba Teradata ODBC Driver DSN Setup dialog box on a Windows machine, you can enter your Teradata Wallet reference string directly in the Teradata Wallet String field. Otherwise, to pass in a reference string to the driver, you must set the `EnableWallet` property to `1` and use the following syntax in place of a password value, where *[WalletString]* is your reference string:

```
$tdwallet([WalletString])
```

For example, the following is a connection string that authenticates the connection using a reference string:

```
Driver=Simba Teradata ODBC Driver;DBCName=192.168.222.160;
UID=jsmith;PWD=$tdwallet(jsmith_wallet_
string);EnableWallet=1;
```

LOB Retrieval Modes

Some Teradata Database instances contain Large Object (LOB) data types, such as BLOB (Binary Large Object) and CLOB (Character Large Object). The Simba Teradata ODBC Driver supports two ways of retrieving LOBs: Deferred Mode and Smart LOB (SLOB) Mode. You can optimize driver performance by configuring the appropriate retrieval mode.

- In Deferred Mode, the driver sends an additional query to retrieve each LOB. By default, the driver uses Deferred Mode.
- In SLOB Mode, the driver retrieves LOBs without sending any additional queries, but may need to cache some LOBs in memory.

To optimize driver performance, use Deferred Mode when retrieving large LOBs that you do not want to cache into memory, and use SLOB Mode when you need to retrieve many small LOBs and want to avoid sending a large number of queries. For example, SLOB Mode can improve driver performance when retrieving geospatial data.

! Important:

If SLOB Mode is not configured properly, it can decrease driver performance instead of improving it.

SLOB Mode Usage Guidelines

SLOB Mode is applicable only when certain size restrictions are met:

- The LOB to be retrieved must be smaller than the size specified by the Max Single LOB Bytes (or `MaxSingleLOBBytes`) setting. The driver falls back to using Deferred Mode when retrieving LOBs that exceed this size.
- If the total amount of LOB data being retrieved from a row exceeds the size specified by the Max Total LOB Bytes Per Row (or `MaxTotalLOBBytesPerRow`) setting, then, after using SLOB Mode to retrieve LOBs up to this size limit, the driver uses Deferred Mode to retrieve the remaining LOBs from that row.

Before enabling SLOB Mode, be aware of the following:

- Do not enable the Use Sequential Retrieval Only option (or the `UseSequentialRetrievalOnly` property) if there is any possibility that you might retrieve LOBs from columns in a non-sequential order. For instance, do not enable this option and then execute a query that retrieves LOBs from the third column in a table, then from the first column, and then from the fifth column. If you enable this option and then retrieve LOBs non-sequentially, the driver discards the LOBs that are returned through SLOB Mode and must then retrieve them all again using Deferred Mode.
- When the Use Sequential Retrieval Only option (or the `UseSequentialRetrievalOnly` property) is disabled, the driver caches the other LOBs that it reads while looking for the one to be retrieved. Caching large amounts of data in memory can decrease performance. To prevent this problem, set the size limits so that the driver does not apply SLOB mode to large LOBs. LOB values that do not meet the requirements for SLOB Mode are retrieved using Deferred Mode instead, and therefore do not get cached.

Controlling the Scope of SLOB Mode Settings

You can configure the settings for SLOB Mode on the connection level or on the statement level. Because the optimal settings vary depending on the size of the specific LOBs that you are retrieving, it may be useful to adjust the settings for each statement as you work with your data.

To configure settings for SLOB Mode on the connection level, specify the relevant driver options in a DSN or connection string. These settings apply to all queries and operations that are executed within the connection. For detailed information about the driver options related to SLOB Mode, see the following:

- Max Single LOB Bytes on page 62
- Max Total LOB Bytes Per Row on page 63
- Use Sequential Retrieval Only on page 73

You can override connection-level settings by using statement attributes. To configure settings for SLOB Mode on the statement level, set the following statement attributes:

- **SQL_ATTR_MAX_SINGLE_LOB_BYTES**: Use this attribute to specify the maximum size of the LOBs (in bytes) that the driver can retrieve using SLOB Mode. LOBs that exceed this size are retrieved using Deferred Mode instead. This attribute corresponds to the Max Single LOB Bytes (or `MaxSingleLOBBytes`) driver option.
- **SQL_ATTR_MAX_LOB_BYTES_PER_ROW**: Use this attribute to specify the maximum size of LOB data per row (in bytes) that the driver can retrieve using SLOB Mode. If the total amount of LOB data contained in a row exceeds this size, then the driver retrieves the LOBs from that row using Deferred Mode instead. This attribute corresponds to the Max Total LOB Bytes Per Row (or `MaxTotalLOBBytesPerRow`) driver option.
- **SQL_ATTR_USE_SEQUENTIAL_RETRIEVAL_ONLY**: Use this attribute to indicate whether you are retrieving LOB data from columns in sequential order. This attribute corresponds to the Use Sequential Retrieval Only (or the `UseSequentialRetrievalOnly`) driver option.

Scalar Function Support

The Simba Teradata ODBC Driver includes full support for all of the scalar functions that are supported by the Teradata Database instance that you connect to. The version of the Teradata Database instance determines which specific scalar functions you can call.

For a list of the scalar functions that are supported by your version of Teradata Database, see the *SQL Functions, Operators, Expressions, and Predicates* book from the Teradata Database documentation set.

When calling a scalar function, it is recommended that you place the function inside an ODBC escape sequence, as this prompts the driver to check if the scalar function is valid before attempting to call it. For example:

```
SELECT {fn MOD(x, y) }
```

For more information about calling scalar functions, see "Scalar Functions" in the *ODBC Driver for Teradata User Guide*.

Special Query Syntax

The Simba Teradata ODBC Driver includes support for SET TRANSFORM GROUP FOR TYPE statements when connected to a Teradata Database instance that also supports this syntax. This DDL statement enables you to specify the transform group to use on Teradata complex data types (CDTs) that support multiple transform groups on the session level.

Typically, to specify a transform that you want to use for one of these CDTs, you would have to create a user account with the transform settings defined. The SET TRANSFORM GROUP FOR TYPE statement enables you to use the appropriate transform without having to create an additional user account. You can execute this statement multiple times to change transform groups during the same session, if needed.

For detailed information about how to write and execute the SET TRANSFORM GROUP FOR TYPE statement, see "SET TRANSFORM GROUP FOR TYPE Statement" in the *ODBC Driver for Teradata User Guide*.

! **Important:**

The SET TRANSFORM GROUP FOR TYPE statement must be executed before the preparation of the main query or after the execution of the main query. If you execute this statement during any other stage of the main query, the driver returns the following error message:

```
Error occurred as a SET TRANSFORM GROUP FOR TYPE
statement was executed between PREPARE and EXECUTE.
```

Driver Configuration Options

Driver Configuration Options lists the configuration options available in the Simba Teradata ODBC Driver alphabetically by field or button label. Options having only key names, that is, not appearing in the user interface of the driver, 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 Teradata ODBC Driver DSN Setup
- Simba Teradata ODBC Driver Options
- Advanced Options
- Logging Options

When using a connection string, 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 Teradata ODBC Driver, or via the key name when using a connection string or configuring a connection from a non-Windows machine:

- Account String on page 54
- Authentication Parameter on page 55
- Data Source DNS Entries on page 55
- Date Time Format on page 56
- Default Database on page 56
- Disable Parsing on page 57
- Enable Custom Catalog Mode For 2.x Applications on page 57
- Enable Data Encryption on page 57
- Enable Extended Statement Info on page 58
- Enable Read Ahead on page 58
- Procedure With SPL Source on page 66
- Retry System Calls (EINTR) on page 66
- Return Empty String In CREATE_ PARAMS Column For SQL_ TIMESTAMP on page 67
- Return Generated Keys on page 67
- Return Max CHAR/VARCHAR Length As 32k on page 68
- Session Character Set on page 68
- Session Mode on page 69
- TDMST Port Number on page 70
- Translation DLL Name on page 70

- Ignore Search Patterns on page 59
- Log Error Events on page 59
- Log Level on page 60
- Log Path on page 61
- Login Timeout on page 61
- Max File Size on page 61
- Max Number Files on page 62
- Max Single LOB Bytes on page 62
- Max Total LOB Bytes Per Row on page 63
- Maximum Response Buffer on page 63
- Mechanism on page 64
- Name or IP Address on page 64
- No HELP DATABASE on page 65
- Password or Teradata Wallet String on page 65
- Procedure With Print Stmt on page 66
- Translation Option on page 70
- UPT Mode on page 71
- Use Column Names on page 71
- Use DATE Data for TIMESTAMP Parameters on page 72
- Use Integrated Security on page 72
- Use NULL For Catalog Name on page 73
- Use Regional Settings for Decimal Symbol on page 73
- Use Sequential Retrieval Only on page 73
- Use TCP_NODELAY on page 74
- Use X Views on page 75
- Username on page 75

Account String

Key Name	Default Value	Required
`AccountString`	The account string that is associated with the specified user name.	No

Description

The account string to use when logging in to the database.

Authentication Parameter

Key Name	Default Value	Required
`AuthenticationParameter` OR `MechanismKey`	None	Yes, if authenticating using a JWT.

Description

Additional parameters that you might need to specify for authentication, depending on the selected authentication mechanism and the database configuration. For example, if a profile is required for authentication, then you can specify your profile as a parameter.

As another example, if you are authenticating using a JSON web token (JWT), then you must specify the token as a parameter. In this case, you would specify the following parameter, where *[JWT_Token]* is the JSON web token that you obtained from the UDA User Service:

```
token=[JWT_Token]
```

Typically, you do not need to specify any parameters to successfully authenticate your connection.

If the parameter contains any of the following special characters, enclose them in braces (`{}`): `* @ [] {} , = ! () ? ;`

For example, when specifying at JWT in the odbc.ini file or in a connection string, you would type the following:

```
AuthenticationParameter={token=[JWT_Token]}
```

Data Source DNS Entries

Key Name	Default Value	Required
`DataSourceDNSEntries`	None	No

Description

This option specifies how the driver determines which DNS entry to connect to.

- If this option is not set, the driver resolves DNS entries dynamically.
- If this option is set to `0`, the driver uses DNS lookup.
- If this option is set to a non-zero value, then the driver uses that number of DNS entries in a round-robin fashion.

Date Time Format

Key Name	Default Value	Required
`DateTimeFormat`	`AAA`	No

Description

This option specifies the format that the driver uses for DATE values when communicating with the database.

- `AAA`: The driver uses ANSI format for DATE values.
- `IAA`: The driver uses Integer format for DATE values.

Default Database

Key Name	Default Value	Required
`DefaultDatabase`	The default database that is associated with the specified user name.	No

Description

The name of the database to access by default.

If this option is not set, then the driver uses the default database assigned to the specified user name.

If a table owner is not specified, then all catalog functions are associated with the default database.

Disable Parsing

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

Description

This option specifies whether the driver parses SQL statements or passes the statements through to the database without making any modifications.

- Enabled (`1`): SQL statements are passed through to the Teradata Database without any modifications.
- Disabled (`0`): SQL statements are parsed by the driver.

Enable Custom Catalog Mode For 2.x Applications

Key Name	Default Value	Required
`Use2xAppCustomCatalogMode`	Clear (`0`)	No

Description

This option provides backwards compatibility for ODBC 2.x applications that use noncompliant search patterns.

Earlier versions of the driver allowed users to create search patterns other than the `%` search pattern stated in the ODBC Programmer's Reference specification. On noncompliant systems, if a NULL value is passed to the SQLTables API for the SchemaName argument, the result is a search for tables by userid, DBC, and default database schema names, rather than the `%` search pattern.

- Enabled (`1`): The driver allows searches by userid, DBC, and default database schema names.
- Disabled (`0`): The driver uses the `%` search pattern.

Enable Data Encryption

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

Description

This option specifies whether the driver encrypts all communication with the database or authentication information only.

- Enabled (`1`): The driver encrypts all data that is passed between the driver and the database.
- Disabled (`0`): The driver encrypts authentication information only.

Enable Extended Statement Info

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

Description

This option specifies whether extended statement information is used when it is available from the database (Teradata Database versions V2R6.2 and later).

- Enabled (`1`): Extended statement information is requested and used, and the ODBC API function SQLDescribeParam is supported.
- Disabled (`0`): Extended statement information is not used, and the ODBC API function SQLDescribeParam is not supported.

Enable Read Ahead

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

Description

This option specifies whether to request the next response message while the current message is being processed.

- Enabled (`1`): The driver requests the next response message while the current message is being processed.
- Disabled (`0`): The driver does not request the next response message until the current message has been processed.

Ignore Search Patterns

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

Description

This option specifies whether the underscore (_) and percent sign (`%`) characters are parsed as normal characters or as search wildcards.

- Enabled (`1`): The underscore (_) and percent sign (`%`) characters are parsed as normal characters.
- Disabled (`0`): The underscore (_) and percent sign (`%`) characters are parsed as ODBC search wildcards.

Log Error Events

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

Description

This option specifies whether the driver logs information to the Event Viewer of the Teradata server.

- Enabled (`1`): Error events are logged to the Event Viewer.
- Disabled (`0`): Error events are not logged to the Event Viewer.

Note:

- This option is available only in the Windows driver.
- This option is a driver-wide configuration option, so its setting applies to all connections that use the Simba Teradata ODBC Driver, and it cannot be set as a connection string property.

Log Level

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

Description

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

! Important:

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

Set the property to one of the following values:

- OFF (`0`): Disable all logging.
- FATAL (`1`): Logs severe error events that lead the driver to abort.
- ERROR (`2`): Logs error events that might allow the driver 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 driver.
- DEBUG (`5`): Logs detailed information that is useful for debugging the driver.
- TRACE (`6`): Logs all driver activity.

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

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

Log Path

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

Description

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

! Important:

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

Login Timeout

Key Name	Default Value	Required
`LoginTimeout`	`20`	No

Description

The number of seconds that the driver waits for a response when logging in to the database.

Max File Size

Key Name	Default Value	Required
`LogFileSize`	`20`	No

Description

The maximum size of each log file in megabytes (MB). After the maximum file size is reached, the driver creates a new file and continues logging.

! Important:

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

Max Number Files

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

Description

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

! Important:

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

Max Single LOB Bytes

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

Description

The maximum size of the LOBs (in bytes) that the driver can retrieve using Smart LOB (SLOB) Mode. LOBs that exceed this size are retrieved using Deferred Mode instead.

If this option is set to `0`, SLOB Mode is disabled, and the driver retrieves all LOB data using Deferred Mode. For more information, see LOB Retrieval Modes on page 49.

Note:

As an alternative to using this option, you can specify this setting on the statement level rather than the connection level by using the SQL_ATTR_MAX_SINGLE_LOB_BYTES statement attribute.

Max Total LOB Bytes Per Row

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

Description

The maximum size of LOB data per row (in bytes) that the driver can retrieve using Smart LOB (SLOB) Mode. If the total amount of LOB data contained in a row exceeds this size, then the driver retrieves the LOBs from that row using Deferred Mode instead.

If this option is set to `0`, SLOB Mode is disabled, and the driver retrieves all LOB data using Deferred Mode. For more information, see LOB Retrieval Modes on page 49.

Note:

As an alternative to using this option, you can specify this setting on the statement level rather than the connection level by using the SQL_ATTR_MAX_LOB_BYTES_PER_ROW statement attribute.

Maximum Response Buffer

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

Description

The maximum size of the response buffer for SQL requests, in kilobytes.

When you are connected to a database instance that is running Teradata Database 16.00 or later, the maximum value is 7361536. For connections that use earlier versions of Teradata Database, the maximum value is 1048576.

Mechanism

Key Name	Default Value	Required
`MechanismName`	None	No

Description

The mechanism that the driver uses to authenticate the connection to the database.

Select one of the following settings, or set the key to the name of the authentication mechanism:

- **KRB5**: The driver uses the Kerberos protocol. The application provides the user name and password.
- **LDAP**: The driver uses the LDAP protocol. The application provides the user name and password.
- **TD2**: The driver uses the Teradata 2 mechanism, which requires you to provide a Teradata Database user name and password. For information about the options that you use to specify your credentials, see Username on page 75 and Password or Teradata Wallet String on page 65.
- **TDNEGO**: The driver uses the mechanism that is selected automatically through Teradata Negotiating, which can include single sign-on.
- **JWT**: The driver uses a JSON web token (JWT) to authenticate the connection. For information about specifying a JWT, see Authentication Parameter on page 55.

Note:

If this option is not set, then the driver uses the authentication mechanism specified in the `tdgssconfigure.xml` file in the TeraGSS program. This is typically TD2.

Name or IP Address

Key Name	Default Value	Required
`DBCName`	None	Yes

Description

The fully qualified domain name or IP address of the Teradata Database instance.

No HELP DATABASE

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

Description

This option specifies whether the Help Database is used.

- Enabled (`1`): SQLTables uses a SELECT statement when no wildcard characters are used in SQLTables.
- Disabled (`0`): The driver uses the HELP DATABASE command.

Note:

If this option is enabled, then SQLTables uses either `dbc.tables` or `dbc.tablesX`, depending on whether X Views are enabled. For more information about X Views, see Use X Views on page 75.

Password or Teradata Wallet String

Key Name	Default Value	Required
`Password`	None	Yes, if the authentication mechanism is TD2.

Description

The password that you use to access the database.

Alternatively, if the Teradata Wallet utility is set up on your machine, you can specify a Teradata Wallet reference string in place of a password. When providing the reference string in the Simba Teradata ODBC Driver DSN Setup dialog box, type the string normally. When providing the reference string in a connection string, you must include the `$tdwallet()` token. For example:

```
Password=$tdwallet([WalletString]);
```

Procedure With Print Stmt

Key Name	Default Value	Required
`PrintOption`	`N`	No

Description

This option specifies whether to enable the print option for stored procedures.

- `P`: The SPL PRINT statements specified in the stored procedure body are saved in the compiled stored procedure.
- `N`: The SPL PRINT statements are not saved. If the Procedure With SPL Source option (the `SplOption` property) is enabled, then the driver preserves the SPL PRINT statements in the SPL source text.

Procedure With SPL Source

Key Name	Default Value	Required
`SplOption`	`Y`	No

Description

This option specifies whether to use stored procedure language (SPL) when creating stored procedures.

- `Y`: SPL is enabled, and the source text must be stored in Teradata Database.
- `N`: SPL is disabled, and the source text is not stored in the server.

Retry System Calls (EINTR)

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

Description

This option specifies whether the driver retries the socket system calls or returns a SQL_ERROR when an EINTR error occurs.

- Enabled (`1`): The driver retries the socket system calls.
- Disabled (`0`): The driver returns a SQL_ERROR, and the ODBC application becomes responsible for recovering from the interrupted socket system calls.

Return Empty String In CREATE_PARAMS Column For SQL_ TIMESTAMP

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

Description

This option specifies whether the driver returns an empty string or the given value for the CREATE_PARAMS column when you call SQLGetTypeInfo for SQL_ TIMESTAMP data.

- Enabled (`1`): The driver returns an empty string, and prohibits Microsoft Access from using any TIMESTAMP precision values when creating tables.
- Disabled (`0`): The driver returns the given value.

Note:

This option is applicable only for Windows and macOS.

Return Generated Keys

Key Name	Default Value	Required
`ReturnGeneratedKeys`	`N`	No

Description

This option determines the result from requests that insert data into identity columns. These requests can optionally return a result set containing identity column values, also known as auto-generated keys, for the inserted rows.

- `C`: The driver retrieves the identity columns only.
- `R`: The driver retrieves the entire row.
- `N`: The driver does not retrieve auto-generated keys.

Return Max CHAR/VARCHAR Length As 32k

Key Name	Default Value	Required
`Use32kMaxCharColumnSize`	Clear (`0`)	No

Description

This option specifies whether the driver returns a hard-coded value for the COLUMN_SIZE column when you call SQLGetTypeInfo for SQL_CHAR and SQL_VARCHAR data. Enabling this option prevents the returned column size from causing numeric overflows in Microsoft Access.

- Enabled (`1`): The driver returns a hard-coded value for the maximum size of SQL_CHAR and SQL_VARCHAR columns.
- Disabled (`0`): The driver returns the actual maximum size of the column. In some cases, Microsoft Access might experience numeric overflow when processing the column size returned by the driver.

Depending on the Session Character Set (or `CharacterSet`) setting, the hard-coded value is 32000 or 64000. For more information, see Session Character Set on page 68.

Note:

This option is applicable only for Windows and macOS.

Session Character Set

Key Name	Default Value	Required
`CharacterSet`	`ASCII`	No

Description

The character set to use for the session. This value can be a user-defined character set, or one of the following pre-defined character sets:

- `ASCII`
- `UTF8`
- `UTF16`
- `LATIN1252_0A`
- `LATIN9_0A`
- `LATIN1_0A`
- `Shift-JIS` (Windows, DOS compatible, `KANJISJIS_0S`)
- `EUC` (Unix compatible, `KANJIEC_0U`)
- `IBM Mainframe` (`KANJIEBCDIC5035_0I`)
- `KANJI932_1S0`
- `BIG5` (`TCHBIG5_1R0`)
- `GB` (`SCHGB2312_1T0`)
- `SCHINESE936_6R0`
- `TCHINESE950_8R0`
- `NetworkKorean` (`HANGULKSC5601_2R4`)
- `HANGUL949_7R0`
- `ARABIC1256_6A0`
- `CYRILLIC1251_2A0`
- `HEBREW1255_5A0`
- `LATIN1250_1A0`
- `LATIN1254_7A0`
- `LATIN1258_8A0`
- `THAI874_4A0`

Note:

The specified character set must be installed on Teradata Database.

Session Mode

Key Name	Default Value	Required
`SessionMode`	`System Default`	No

Description

This option specifies the session mode that the driver uses during sessions on the database.

- `ANSI`: The driver uses ANSI mode.
- `System Default`: The driver uses the default session mode of the system that you are using the driver on.
- `Teradata`: The driver uses Teradata mode.

TDMST Port Number

Key Name	Default Value	Required
`TdmstPortNumber`	`1025`	No

Description

The number of the port used to access Teradata Database.

> ! **Important:**
>
> Do not change this value unless instructed to do so by Technical Support.

Translation DLL Name

Key Name	Default Value	Required
`TranslationDllName`	None	No

Description

The full path to the `.dll` file that contains functions for translating all the data that is transferred between the Teradata server and the driver.

This `.dll` file is used for translation if local character sets are not supported by Teradata Database or the driver.

Translation Option

Key Name	Default Value	Required
`TranslationOption`	None	No

Description

The options used by the Translation DLL file (see Translation DLL Name on page 70). The required options may vary depending on the Translation DLL file being used.

UPT Mode

Key Name	Default Value	Required
`UPTMode`	Notset (`NOTSET`)	No

Description

This option specifies whether the driver supports Unicode Pass Through (UPT) for Pass Through Characters (PTCs). For more information about UPT, see "Unicode Pass Through" in the Teradata Database documentation: http://info.teradata.com/htmlpubs/DB_TTU_16_00/index.html#page/General_Reference/B035-1098-160K/ifk1472240714022.html.

- Notset (`NOTSET`): The driver does not do anything to change UPT support.
- UPTON (`UPTON`): The driver sends a query to the database to enable UPT support. When UPT support is enabled, the driver allows PTCs to be passed through to the database.
- UPTOFF (`UPTOFF`): The driver sends a query to the database to disable UPT support. When UPT support is disabled, the driver does not allow PTCs to be passed through to the database.

Use Column Names

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

Description

This option specifies whether column names or column titles are returned.

- Enabled (`1`): The driver returns column names.
- Disabled (`0`): The driver returns column titles if they are defined. Otherwise, the driver returns column names.

Note:

Column titles for SQLColumns are shown in the LABEL column.

Use DATE Data for TIMESTAMP Parameters

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

Description

This option specifies whether the driver sends DATE data for parameters that are bound as SQL_TIMESTAMP or SQL_C_TIMESTAMP.

Important:

This option should only be enabled for applications that use Microsoft Access Jet databases, as it can result in truncating SQL_C_TIMESTAMP data.

- Enabled (`1`): The driver sends DATE data for SQL_TIMESTAMP and SQL_C_TIMESTAMP parameters.
- Disabled (`0`): The driver sends standard data for these parameters.

Use Integrated Security

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

Description

This option specifies whether the driver authenticates the connection using Single Sign-On (SSO) or Conventional Sign-On (CSO).

- Enabled (`1`): The driver uses SSO and authenticates the connection by using Teradata Database credentials that are derived from the user information on your client machine.
- Disabled (`0`): The driver uses CSO and requires you to provide your Teradata Database credentials.

Use NULL For Catalog Name

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

Description

This option specifies whether the driver sets any Catalog Name parameters to NULL.

- Enabled (`1`): Catalog Name parameters are set to NULL for all Catalog API functions, even if the application passes a value.
- Disabled (`0`): Catalog Name parameter values are passed in. In this case the driver returns an error, because Teradata Database does not support catalogs.

Use Regional Settings for Decimal Symbol

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

Description

This option specifies whether the driver uses the regional settings for decimal symbols, or uses a period (`.`) regardless of the regional settings.

- Enabled (`1`): The driver uses the regional settings for decimal symbols.
- Disabled (`0`): The driver uses a period (`.`) for decimal symbols regardless of the regional settings.

Note:

This option is applicable only for Windows and macOS.

Use Sequential Retrieval Only

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

Description

This option indicates to the driver whether you are retrieving LOB data from columns in sequential order or non-sequential order. When working in Smart LOB (SLOB) Mode, the driver reads and caches LOB data differently depending on this setting. For more information about SLOB Mode, see LOB Retrieval Modes on page 49.

- Enabled (`1`): When working in SLOB Mode, the driver does not cache the other LOBs that it reads while looking for the one to be retrieved. Because the driver can retrieve LOBs in a single pass if they are queried sequentially, the driver does not need to cache them.
- Disabled (`0`): When working in SLOB Mode, the driver caches the other LOBs that it reads while looking for the one to be retrieved. This caching allows the driver to successfully retrieve SLOBs in any order.

! Important:

- Do not enable this option if there is any possibility that you might retrieve LOBs from columns in a non-sequential order. For instance, do not enable this option and then execute a query that retrieves LOBs from the third column in a table, then from the first column, and then from the fifth column. If you enable this option and then retrieve LOBs non-sequentially, the driver discards the LOBs that are returned through SLOB Mode and must then retrieve them all again using Deferred Mode.
- As an alternative to using this option, you can specify this setting on the statement level rather than the connection level by using the SQL_ATTR_ USE_SEQUENTIAL_RETRIEVAL_ONLY statement attribute.

Use TCP_NODELAY

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

Description

This option specifies whether TCP immediately sends small packets or waits to gather packets into a single, larger packet.

- Enabled (`1`): TCP immediately sends small packets. This option can avoid transmission delays but might increase network traffic.
- Disabled (`0`): TCP gathers small packets into a single larger packet. This option can reduce network traffic but might cause transmission delays.

Use X Views

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

Description

This option specifies whether to use X views. X views restrict access to the data so that the driver can only access objects that the specified user owns or controls.

- Enabled (`1`): The driver uses the following views:
  - SQLTables() and SQLProcedures() use `dbc.tablesVX` and `dbc.databasesVX`
  - SQLColumns() and SQLProcedureColumns() use `dbc.columnsVX`
  - SqlStatistics() uses `dbc.tablesizeVX`
- Disabled (`0`): The driver uses the following views:
  - SQLTables() and SQLProcedures() use `dbc.tablesV` and `dbc.databasesV`
  - SQLColumns() and SQLProcedureColumns() use `dbc.columnsV`
  - SqlStatistics() uses `dbc.tablesizeV`

Username

Key Name	Default Value	Required

`UID`
OR
`Username`
	None	Yes, if the authentication mechanism is TD2.

Description

Your user name for authenticating the connection to Teradata Database through the specified authentication mechanism. For example, if you set the **Mechanism** option to **TD2** (set the `MechanismName` key to `TD2`), then you must provide your Teradata Database user name.

Configuration Options Having Only Key Names

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

DataSourceName / DSN

Key Name	Default Value	Required

`DataSourceName`
OR
`DSN`
	None	No

Description

The name of the DSN that you want to use to connect to Teradata Database.

Note:

This property is used in connection strings only. It cannot be set in the `odbc.ini` file.

Driver

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

Description

On Windows, the name of the installed driver (`Simba Teradata ODBC Driver`).

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

DriverLocale

Key Name	Default Value	Required
`DriverLocale`	`en-US`	No

Description

The locale to use for error messages.

Set this property to one of the following values:

- `en-US`: The driver returns error messages in English.
- `ja-JP`: The driver returns error messages in Japanese.

EnableWallet

Key Name	Default Value	Required
`EnableWallet`	`0`	Yes, if using a Teradata Wallet reference string instead of a password.

Description

This option specifies whether the driver authenticates the connection using a Teradata Wallet reference string instead of a password.

- `1`: The driver uses a Teradata Wallet reference string.
- `0`: The driver uses a password.

For more information, see Teradata Wallet on page 48 and Password or Teradata Wallet String on page 65.

IANAAppCodePage

Key Name	Default Value	Required
`IANAAppCodePage`	None	No

Description

The ODBC application code page that the driver uses when converting characters between ANSI and Unicode.

For a list of supported values, see "ODBC Application Code Page Values" in Teradata's *ODBC Driver for Teradata User Guide*.

Note:

- This property is applicable only for macOS and Linux.
- This setting takes precedence over the `CharacterSet` setting. For information about the `CharacterSet` setting, see Session Character Set on page 68.

Third-Party Trademarks

Third-Party Licenses

The licenses for the third-party libraries that are included in this product are listed below.

CityHash License

CityHash, by Geoff Pike and Jyrki Alakuijala

http://code.google.com/p/cityhash/

dtoa License

The author of this software is David M. Gay.

THE MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.

Expat License

Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NOINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

ICU License - ICU 1.8.1 and later

COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1995-2014 International Business Machines Corporation and others

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES,

OpenSSL License

Original SSLeay License

Stringencoders License

Teradata Tools and Utilities License

either Teradata U.S., Inc. for Software deliveries in the US or Teradata Ireland Ltd. for Software deliveries outside the United States. "Software" refers to the software product identified above, which consists of computer software code in object code form only, as well as associated documentation that Teradata may elect in its sole discretion to provide you. "Software" also includes any and all error corrections, bug fixes, updates, upgrades, or new versions or releases of the Software (collectively and individually, "Enhancements") that Teradata may elect in its sole discretion to provide you.

"Scenario (i)": If you are downloading the Software on behalf of a Teradata-authorized licensee that has already purchased a license to the Software pursuant to an executed master agreement ("Master Agreement") with Teradata or one of its affiliates, the terms of such master agreement prevail over this Agreement except for Section 2.a.(i). If you have only purchased a license to some of the tools and utilities in the Software ("Purchased Utilities"), you are not authorized to use other tools and utilities in the Software for which you have not yet purchased a license ("Un-purchased Utilities") simply by virtue of their inclusion in the Software. You may also be required to purchase an upgrade to the Purchased Utilities in conjunction with an upgrade to your purchased license to the Teradata Relational Database product.

"Scenario (ii)": If Scenario (i) does not apply to you, then this Agreement and the terms of use for the site from which you downloaded the Software ("General Terms of Use") constitute the entire understanding of the parties with respect to the Software and Services, and supersede all other prior agreements and understandings whether oral or written. For clarity, Section 2.a.(i) does not apply to you. For example, individuals that have downloaded a copy of the Teradata Express version of the Teradata Relational Database product fall within Scenario (ii).

1. **Term**. This Agreement commences on the earliest date of the first download, first copying, first installation, or first use of the Software (the "Effective Date"). Unless terminated earlier as provided herein, this agreement, including your license to the Software, will expire or terminate on the same date that your Teradata-authorized license to use the Teradata Relational Database product expires or terminates (whichever occurs first).
2. **License**.

   (a) Teradata grants you a nonexclusive, nontransferable, paid up license:

   (i) If **Scenario (i)** applies - subject to your compliance with all of the terms and conditions of the Master Agreement, to install and use the Purchased Utilities on any number and type of computers in object code form for your internal use solely for the purpose of facilitating your Teradata-authorized license to use the Teradata Relational Database product. You may use the Un-purchased Utilities for up to ninety (90) days solely for purposes of internally evaluating whether to purchase a license to the Un-purchased Utilities.

(ii) If **Scenario (ii)** applies - subject to your compliance with all of the terms and conditions of this Agreement, to install and use the Software on your computer workstation in object code form for your internal use solely for purposes of facilitating your Teradata-authorized license to use the Teradata Relational Database product. You may make reasonable archival backup copies of the Software, but may only use an archival copy in lieu of your primary copy and subject to the same restrictions as your primary copy.

(b) The term Third Party Software means computer programs or modules (including their documentation) that bear the logo, copyright and/or trademark of a third party (including open source software that are contained in files marked as "open source" or the like) or are otherwise subject to written license terms. Third Party Software does not constitute Software. Third Party Software is licensed to you subject to the applicable license terms accompanying it, included in/with it, referenced in it, or otherwise entered into by you with respect to it. Third Party Software license terms include those found in the FOSS licensing zip file accompanying the Software. Teradata provides source code to certain Third Party Software for certain periods of time in compliance with certain applicable licenses. To request such source code, visit http://developer.teradata.com/download/license/oss-request.

(c) You will not sell, copy, rent, loan, modify, transfer, disclose, embed, sublicense, create derivative works of or distribute the Software, in whole or in part, without Teradata’s prior written consent. You are granted no rights to obtain or use the Software’s source code. You will not reverse-assemble, reverse compile or reverse-engineer the Software, except as expressly permitted by applicable law without the possibility of contractual waiver. Notwithstanding anything to the contrary, you do not have any license, right, or authority to subject the Software, in whole or in part or as part of a larger work, to any terms of any other agreement, including GNU Public Licenses.

(d) No license rights to the Software will be implied. The Software, which includes all copies thereof (whether in whole or in part), is and remains the exclusive property of Teradata and its licensors. You will ensure that all copies of the Software contain Teradata's and its licensors' copyright notices, as well as all other proprietary legends. Teradata reserves the right to inspect your use of the Software for purposes of verifying your compliance with the terms and conditions of this Agreement.

3. **Responsibilities**. You are responsible for the installation of the Software, as well as for providing data security and backup operations. This Agreement does not require Teradata to provide you with any Enhancements, consulting services, technical assistance, installation, training, support, or maintenance of any kind (collectively and individually, "Services"). To the extent that Teradata elects to provide you with any Services, such Services are provided to you at Teradata's sole discretion and may be modified or discontinued at any time for any reason.
4. **DISCLAIMER OF WARRANTY**. TERADATA: (a) PROVIDES SERVICES (IF ANY), (b) LICENSES THE SOFTWARE, AND (c) PROVIDES THIRD PARTY SOFTWARE TO YOU ON AN "AS-IS" BASIS WITHOUT WARRANTIES OF ANY KIND (ORAL OR WRITTEN, EXPRESS OR IMPLIED, OR STATUTORY). WITHOUT LIMITATION TO THE FOREGOING, THERE ARE NO IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. TERADATA DOES NOT WARRANT THAT THE SOFTWARE, THIRD PARTY SOFTWARE, OR SERVICES WILL MEET YOUR REQUIREMENTS OR CONFORM TO ANY SPECIFICATIONS, OR THAT THE OPERATION OF THE SOFTWARE OR THIRD PARTY SOFTWARE WILL BE UNINTERRUPTED OR ERROR FREE. YOU BEAR THE ENTIRE RISK AS TO SATISFACTORY QUALITY, PERFORMANCE, ACCURACY, AND RESULTS OBTAINED FROM THE SOFTWARE, THIRD PARTY SOFTWARE, AND SERVICES.

   SOME JURISDICTIONS RESTRICT DISCLAIMERS OF WARRANTY, SO THE ABOVE DISCLAIMERS MAY NOT FULLY APPLY TO YOU.
5. **LIMITATIONS ON LIABILITY**: UNDER NO CIRCUMSTANCES WILL TERADATA'S AND ITS LICENSORS' TOTAL CUMULATIVE LIABILITY FOR CLAIMS RELATING TO THIS AGREEMENT, THE SERVICES, THE SOFTWARE, AND/OR THIRD PARTY SOFTWARE (WHETHER BASED IN CONTRACT, STATUTE, TORT (INCLUDING NEGLIGENCE) OR OTHERWISE) EXCEED US$1,000; PROVIDED, HOWEVER, THAT THE FOREGOING WILL NOT APPLY TO CLAIMS FOR (A) PERSONAL INJURY, INCLUDING DEATH, TO THE EXTENT CAUSED BY TERADATA'S NEGLIGENCE OR WILLFUL MISCONDUCT; OR (B) PHYSICAL DAMAGE TO TANGIBLE REAL OR PERSONAL PROPERTY TO THE EXTENT CAUSED BY TERADATA'S NEGLIGENCE OR WILLFUL MISCONDUCT EQUAL TO THE AMOUNT OF DIRECT DAMAGES UP TO ONE MILLION DOLLARS PER OCCURRENCE. IN NO EVENT WILL TERADATA OR ITS LICENSORS BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL DAMAGES, OR FOR LOSS OF PROFITS, REVENUE, TIME, OPPORTUNITY OR DATA, EVEN IF INFORMED OF THE POSSIBILITY OF SUCH DAMAGES.

SOME JURISDICTIONS RESTRICT LIMITATIONS OF LIABILITY, SO THE ABOVE LIMITATIONS MAY NOT FULLY APPLY TO YOU.

6. **Government Restrictions**. You agree that you will not, directly or indirectly, export or transmit any Software without obtaining Teradata's prior written authorization, as well as appropriate governmental approvals, including those required by the U.S. Government. Use and or distribution of this software is subject to export laws and regulations of the United States and other jurisdictions. The links below connect you to applicable U.S. government agencies, and their regulations, that have jurisdiction over this transaction.

   http://www.bis.doc.gov/

   http://www.treas.gov/offices/enforcement/ofac/

   In downloading this product, you acknowledge that this transaction is subject to applicable export control laws and that your download, use and/or subsequent distribution of this product is not prohibited under applicable laws and regulations.

   Teradata's commercial computer software and commercial computer software documentation is provided to the U.S. Government in accordance with: (a) the Restricted Rights Notice set forth in 48 CFR 52.227-14 (Dec 2007); (b) Teradata's standard commercial license rights supplemented by FAR 52.227-19 (Dec 2007); and/or (c) the limited rights and license set forth 48 CFR 252.227-7015 (Nov 1995), as applicable.
7. **Termination and Expiration**. A party may terminate this Agreement with or without cause, upon providing written notice to the other parties. When this Agreement terminates or expires, you will immediately cease all use of the Software, permanently remove the Software from all computers, destroy all copies of the Software, and (upon receipt of Teradata's request) provide a signed written certification that the foregoing has occurred. Sections 4, 5, 6, 7, 8, 9, 10, and 11 will survive expiration or termination of this Agreement.
8. **Choice of Law and Dispute Resolution**. The parties will attempt in good faith to resolve any controversy or claim by negotiation or mediation. If they are unable to do so, and regardless of the causes of action alleged and whether they arise under this Agreement or otherwise, the claim will be resolved by arbitration before a sole arbitrator in Dayton, Ohio pursuant to the then-current Commercial Rules of the American Arbitration Association and the federal substantive and procedural law of arbitration. The arbitrator's award will be final and binding, and may be entered in any court having jurisdiction thereof, but may include only damages consistent with the limitations in this Agreement. Each party will bear its own attorney's fees and costs related to the arbitration. The obligations to negotiate, mediate and arbitrate shall not apply to claims for misuse or infringement of a party's intellectual property rights. Any claim or action must be brought within two years after the cause of action accrues. New York law will

govern the interpretation and enforcement of this Agreement, except that the Federal Arbitration Act will govern the interpretation and enforcement of the arbitrability of claims under this Section.

9. **Feedback**. Notwithstanding anything to the contrary: (a) Teradata will have no obligation of any kind with respect to any Software-related comments, suggestions, design changes or improvements that you elect to provide to Teradata in either verbal or written form (collectively, "Software Feedback"), and (b) Teradata and its affiliates are hereby free to use any ideas, concepts, know-how or techniques, in whole or in part, contained in Software Feedback: (i) for any purpose whatsoever, including developing, manufacturing, and/or marketing products and/or services incorporating Software Feedback in whole or in part, and (ii) without any restrictions or limitations, including requiring the payment of any license fees, royalties, or other consideration.
10. **Confidentiality**. You will not disclose the results of any testing or evaluations, including any benchmarks, insofar as it relates to the Software without Teradata's prior written consent.
11. **Miscellaneous**. In the event of a conflict between this Agreement and the General Terms of Use, this Agreement will prevail with respect to the subject matter hereof. No oral representation or change to this Agreement will be binding upon either party unless agreed to in writing and signed by authorized representatives of all parties. You will not assign this Agreement or your rights, nor will you delegate your obligations under this Agreement. Failure by either party to enforce any term or condition of this Agreement will not be deemed a waiver of future enforcement of that or any other term or condition. The provisions of this Agreement are severable. "Include", "includes", and "including" shall be interpreted as introducing a list of examples which do not limit the generality of any preceding words or any words in the list of examples.