



Simba Trino ODBC Data Connector

Installation and Configuration Guide

Version 2.3

February 2026

Contents

Contents	2
About This Guide	6
Purpose	6
Audience	6
Knowledge Prerequisites	6
Document Conventions	6
About the Simba Trino ODBC Connector	7
About Trino	7
About the Driver	7
Platform and Data source version support	9
Windows Connector	10
Windows System Requirements	10
Installing the Connector in Windows	10
Creating a Data Source Name in Windows	11
Configuring Advanced Options in Windows	12
Configuring Authentication in Windows	14
Configuring SSL Verification in Windows	18
Configuring a Proxy Connection in Windows	19
Configuring FIPS in Windows	19
Exporting a Data Source Name in Windows	20
Importing a Data Source Name in Windows	20
Configuring Logging Options in Windows	20

Verifying the Connector Version Number in Windows	23
macOS Connector	24
macOS System Requirements	24
Installing the Connector in macOS	24
Verifying the Connector Version Number in macOS	25
Configuring FIPS in mac OS	25
Uninstalling the Connector in macOS	25
Linux Connector	27
Linux System Requirements	27
Installing the Connector Using the RPM File	27
Installing the Connector on Debian	28
Verifying the Connector Version Number in Linux	29
Configuring FIPS in Linux	30
Configuring the ODBC Driver Manager in Non-Windows Machines	31
Specifying ODBC Driver Managers in Non-Windows Machines	31
Specifying the Locations of the Connector Configuration Files	32
Configuring ODBC Connections in Non-Windows Machine	34
Creating a Data Source Name in a Non-Windows Machine	34
Configuring a DSN-less Connection in a Non-Windows Machine	36
Configuring Authentication in a Non-Windows Machine	37
Configuring SSL Verification in a Non-Windows Machine	40
Configuring Logging Options in a Non-Windows Machine	41
Testing the Connection in Non-Windows Machine	42

Using a Connection String	44
DSN Connection String Example	44
DSN-less Connection String Examples	44
Features	46
Catalog and Schema Support	46
Parameters	46
Resource Group	46
Data Types	47
Security and Authentication	48
Connector Configuration Options	49
Configuration Options Appearing in the User Interface	50
Configuration Options Having Only Key Names	72
Third-Party Trademarks	75

Copyright ©2026 Simba Technologies Inc., an insightsoftware company. All Rights Reserved.

Information in this document is subject to change without notice. Companies, names and data used in examples herein are fictitious unless otherwise noted. No part of this publication, or the software it describes, may be reproduced, transmitted, transcribed, stored in a retrieval system, decompiled, disassembled, reverse-engineered, or translated into any language in any form by any means for any purpose without the express written permission of Simba Technologies Inc.

Parts of this Program and Documentation include proprietary software and content that is copyrighted and licensed by Simba Technologies Incorporated. This proprietary software and content may include one or more feature, functionality or methodology within the ODBC, JDBC, ADO.NET, OLE DB, ODBO, XMLA, SQL and/or MDX component(s).

For information about Simba's products and services, visit: www.insightsoftware.com.

Contact Us

For support, visit <https://trino.io/foundation>

About This Guide

Purpose

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

Audience

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

Knowledge Prerequisites

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

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

Document Conventions

Italics is used when referring to book and document titles.

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

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



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



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

About the Simba Trino ODBC Connector

About Trino

Trino is a low latency distributed query engine capable of querying large datasets from multiple data sources using SQL. Trino is designed for short, interactive queries useful for data exploration.

The data sources that Trino supports include MySQL and PostgreSQL. Trino also integrates seamlessly with the Hive metastore to complement existing Hive environments with low latency queries. Unlike traditional RDBMS or SQL-on-Hadoop solutions that require centralized schema definitions, Trino can query self-describing data as well as complex or multi-structured data that is commonly seen in big data systems. Moreover, Trino does not require a fully structured schema and can support semi-structured or nested data types such as JSON.

Trino processes the data in record batches and discovers the schema during the processing of each record batch. Thus, Trino has the capability to support changing schemas over the lifetime of a query. Trino reconfigures its operators and handles these situations to ensure that data is not lost.

About the Driver

The Simba Trino ODBC Connector lets organizations connect their BI tools to Trino. Trino provides an ANSI SQL query layer and also exposes the metadata information through an ANSI SQL standard metadata database called INFORMATION_SCHEMA. The Simba Trino ODBC Connector leverages INFORMATION_SCHEMA to expose Trino's metadata to BI tools as needed.

The connector complies with the ODBC 3.80 data standard, including important functionality such as Unicode and 32- and 64-bit support for high-performance computing environments on all platforms.

ODBC is one of the most established and widely supported APIs for connecting to and working with databases. At the heart of the technology is the ODBC connector, which connects an application to the database. For more information about ODBC, see: <https://insightsoftware.com/blog/what-is-odbc/>. For complete information about the ODBC specification, see the *ODBC API Reference* from the Microsoft documentation: <https://docs.microsoft.com/en-us/sql/odbc/reference/syntax/odbc-api-reference>.

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

**Note:**

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

About SQLParse Methods

The SQL_FIRST and SQL_PARSE methods may lead to different behavior for similar queries. This occurs when the connector switches between the two modes trying to find a query language that can support the inputted query. Behavior will be consistent when the query language the connector decides to use remains consistent, however changes to the query may cause it to fail in one of the

languages. For example, when using SOQL_FIRST mode you issue a query that is executable as SOQL. In a subsequent transaction, you slightly modify the query and it becomes invalid in SOQL, but is valid in SQL. This results in the first query being executed using SOQL and the second query being executed using SQL. This can cause slight differences between the two result sets since the behavior is not the same for all SOQL and SQL queries.

One instance where this occurs is comparisons involving null values, because SOQL and SQL handle comparisons against null differently. SQL returns an unknown state if a comparison operator (such as = or >) is used with null, and the results contain zero rows. However, SOQL allows such a comparison and returns results.

Using the SOQL_FIRST mode, you issue the query `SELECT Name FROM Account WHERE NumberOfEmployees = NULL`. This query is valid SOQL and the returned values contain all non-null values as specified by SOQL. Next you issue the query `SELECT Account.Name FROM Account, Contact WHERE Account.Id = Contact.AccountId AND Account.NumberOfEmployees = NULL`. This query is not valid SOQL but is valid SQL. It returns zero values as specified by the SQL specification. The first query may have lead you to believe that the second query would also return results, but difference in query language used means second query returns no results.

If you require consistent behavior in these types of instances, use either SOQL_ONLY or the SQL_ONLY mode.

Platform and Data source version support

The Simba Trino ODBC Connector supports Windows, macOS, and Linux operating systems. For detailed information on supported operating systems and data source versions, please refer to the connector's release notes.

Windows Connector

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

Windows System Requirements

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

- Administrator rights on your machine.
- 75 MB of available disk space

Before the connector can be used, the Visual C++ Redistributable for Visual Studio 2022 with the same bitness as the connector must also be installed. If you obtained the connector from the Simba website, then your installation of the connector automatically includes this dependency. Otherwise, you must install the redistributable manually. You can download the installation packages for the redistributable at <https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-170>.

Installing the Connector in Windows

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

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

- `SimbaTrinoODBC32.msi` for 32-bit applications
- `SimbaTrinoODBC64.msi` for 64-bit applications

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

To install the Simba Trino ODBC Connector in Windows:

1. Depending on the bitness of your client application, double-click to run **SimbaTrinoODBC32.msi** or **SimbaTrinoODBC64.msi**.
2. Click **Next**.
3. Select the check box to accept the terms of the License Agreement if you agree, and then click **Next**.
4. To change the installation location, click **Change**, then browse to the desired folder, and then click **OK**. To accept the installation location, click **Next**.
5. Click **Install**.
6. When the installation completes, click **Finish**.

7. If you received a license file through email, then copy the license file into the `\lib` subfolder of the installation folder you selected above. You must have Administrator privileges when changing the contents of this folder.


Creating a Data Source Name in Windows

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

Alternatively, for information about DSN-less connections, see [Using a Connection String](#).

To create a Data Source Name in Windows:


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

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


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

3. Choose one:

- To create a DSN that only the user currently logged into Windows can use, click the **User DSN** tab.
- Or, to create a DSN that all users who log into Windows can use, click the **System DSN** tab.

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

4. Click **Add**.
5. In the Create New Data Source dialog box, select and then click **Finish**. The DSN Setup dialog box opens.
6. In the **Data Source Name** field, type a name for your DSN.
8. If the database that you are connecting to requires authentication, then use the options in the Authentication area to configure authentication as needed. For more information, see [Configuring Authentication in Windows](#)
9. In the **Host** field, type the IP address or host name of the Trino server.
10. In the **Port** field, type the number of the TCP port that the Trino server uses to listen for client connections.

 **Note:** The default port number used by Trino is 8080.

11. In the **Catalog** field, type the name of the synthetic catalog under which all of the schemas/databases are organized.

12. In the **Schema** field, type the name of the schema for the connector to use.
13. Optionally, in the **Time Zone ID** field, type the name of the time zone for the connector to use, in tz database format. For a list of time zones in tz database format, see https://en.wikipedia.org/wiki/List_of_tz_database_time_zones. If a time zone is not specified, the connector uses the system time zone.
14. To configure client-server verification over SSL, click **SSL Options**. For more information, see [Configuring SSL Verification in Windows](#).
15. To configure advanced connector options, click **Advanced Options**. For more information, see [Configuring Advanced Options in Windows](#).
16. To configure a connection to a datasource through a proxy server, click **Proxy Options**. For more information, see [Configuring a Proxy Connection in Windows](#).
17. To configure logging behavior for the driver, click **Logging Options**. For more information, see [Configuring Logging Options in Windows](#).
18. 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 Trino ODBC Driver DSN Setup dialog box are correct. Contact your Trino server administrator as needed.

19. To save your settings and close the Simba Trino ODBC Driver DSN Setup dialog box, click **OK**.
20. To close the ODBC Data Source Administrator, click **OK**.

Configuring Advanced Options in Windows

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

To configure advanced options in Windows:

1. Open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Advanced Options**.
2. To specify the version of the Trino server that the connector is connecting to, in the **Server Version** field, type the server version number.
3. To automatically test the connection, select **Connection Test**.
4. To automatically populate the metadata for parameters, select **Auto Populate Parameter Metadata**.
5. Choose one:
 - To return SQL_WVARCHAR for VARCHAR columns, and SQL_WCHAR for CHAR columns, select the **Use Unicode SQL Character Types** check box.
 - Or, to return SQL_VARCHAR for VARCHAR columns and SQL_CHAR for CHAR columns, clear the **Use Unicode SQL Character Types** check box.

6. To make calls to SQLTables and SQLColumns without specifying a catalog, select **Allow Metadata From Multiple Catalogs**.



Note: When this option is enabled and the connector makes a call to SQLTables or SQLColumns, the connector queries all catalogs. This may impact performance.

7. To match underscore as a literal in metadata queries, select **Assume Literal Underscore In Metadata Calls**.
8. To use the schema name passed in the DSN for metadata queries, select **Use DSN Schema For Metadata**.
9. To use the System Catalog API to run metadata queries, select **Use System Catalog For Metadata**.
10. To use an equal sign (=) in metadata queries, select **Use Equal In Metadata Filters**.
11. To enable the connector to ignore SQL_ATTR_AUTOCOMMIT and always auto commit, select **AutoCommit Always True**.
12. To allow HTTP redirects, select **Allow HTTP Redirect**.
13. To disable server warnings, select **Mute Server Warnings**.
14. To make sure that the username is in lowercase, enable **Force LowerCase UserName**.
15. To specify the maximum data type length for complex types that the connector casts to VARCHAR (JSON, MAP, ROW, and ARRAY), in the **Max Complex Type Column Length** field, type the maximum length.
16. To specify the maximum number of characters that the connector can return for the names of certain database objects, do one or more of the following:
 - In the **Max Catalog Name Length** field, type the maximum number of characters that the reports for SQL_MAX_CATALOG_NAME_LEN .
 - In the **Max Schema Name Length** field, type the maximum number of characters that the reports for SQL_MAX_SCHEMA_NAME_LEN.
 - In the **Max Table Name Length** field, type the maximum number of characters that the reports for SQL_MAX_TABLE_NAME_LEN.
 - In the **Max Column Name Length** field, type the maximum number of characters that the reports for SQL_MAX_COLUMN_NAME_LEN.
 - In the **Max Varchar Column Length** field, type the maximum number of characters for VARCHAR column names.
 - In the **Max Prepared Statement Length** field, type the maximum prepared query size.

17. To configure the connector to use a Trino Resource Group, do one or more of the following:
 - Optionally, in the **Application Name Prefix** field, type any required prefixes for the Application Name property.
 - In the **Application Name** field, type the application flag you want applied to the queries sent by the connector.
 - In the **ClientTags** field, type a comma-separated list of resource group tags that you want applied to the queries sent by the connector.
18. In the **ExtraCredentials** field, type a comma-separated list of key-value pairs that you want to pass to an external service.
19. In the **ExtraCredentialsDelimiter** field, type a one character delimiter to split up the key-value pairs that you want to pass to an external service.
20. In the **CallingAppName** field, type the name of the application you want to use.
21. In the **Roles** field, type a comma-separated list of key-value pairs for catalog and role.
22. To save your settings and close the Advanced Options dialog box, click **OK**.

Configuring Authentication in Windows

Some Trino data stores require authentication. You can configure the Simba Trino ODBC Connector to provide your credentials and authenticate the connection to the database using one of the following methods:

- [Configuring Kerberos Authentication in Windows](#)
- [Configuring LDAP Authentication in Windows](#)
- [Configuring JWT Authentication in Windows](#)
- [Configuring OIDC Authentication in Windows](#)
- [Configuring AzureAD Authentication in Windows](#)



Note: If Kerberos or LDAP authentication is enabled, then SSL is automatically enabled.

Configuring Kerberos Authentication in Windows

You can configure the connector to use the Kerberos protocol to authenticate the connection.

When you log in to Windows, the operating system automatically caches your credentials. When the connector is run, it loads your Kerberos credentials from the Windows Kerberos cache.

When using Kerberos authentication:

- The connector sends the Kerberos default user principal name as the user name.
- When GSSAPI is enabled (MIT Kerberos) and the Kerberos ticket is generated, the default user principal name is retrieved from the MIT Kerberos credential cache.
- When GSSAPI is disabled (AD Kerberos) and the Kerberos ticket is generated, the default user principal name is retrieved from the Windows Kerberos credential cache.
- If the connector is unable to retrieve the Kerberos default user principal name in either case of MIT or AD Kerberos, the connector sends the default user name `TrinoODBC_Driver`, and reports a warning in the logs.

Note: If Kerberos authentication is enabled, then SSL is automatically enabled.

To configure the connector to use Kerberos authentication in Windows:

1. Open the ODBC Data Source Administrator where you created the DSN, select the DSN, and then click **Configure**. The DSN Setup dialog box opens.
2. From the **Authentication Type** drop-down list, select **Kerberos Authentication**.
3. To use the MIT Kerberos library, select the **Use GSSAPI** check box.
4. Optionally, to generate a ticket using a Kerberos user name and password:
 - a. Select the **Use Existing Kerberos Credentials** check box to use the existing Kerberos Credentials, or clear the check box to generate new credentials.
 - b. Click **Kinit Options**. The Kinit Options dialog box opens.
 - c. From the **Kinit Type** drop-down list, select **Kinit with Password**.
 - d. Optionally, to forward the generated Kerberos credentials, select **Delegate Kerberos Credentials**.
 - e. In the **Kerberos Username** field, type your Kerberos user name.
 - f. In the **Kerberos Password** field, type your Kerberos password.
5. Optionally, to generate a ticket using a Kerberos user name and a keytab file:
 - a. Select the **Use Existing Kerberos Credentials** check box to use the existing Kerberos Credentials, or clear the check box to generate new credentials.
 - b. Click **Kinit Options**. The Kinit Options dialog box opens.
 - c. From the **Kinit Type** drop-down list, select **Kinit with Keytab**.
 - d. Optionally, to forward the generated Kerberos credentials, select **Delegate Kerberos Credentials**.
 - e. In the **Kerberos Username** field, type your Kerberos user name.
 - f. In the **Keytab File Path** field, select the full path of the keytab file.
6. Optionally, to use a service principal name other than the default of HTTP, in the **Service Name** field, type the service name of the Trino server.

7. To configure client-server verification over SSL, click **SSL Options**. For more information, see [Configuring SSL Verification in Windows](#).
8. To save your settings and close the dialog box, click **OK**.

You can now use the connector to authenticate through Kerberos and connect to your Trino server.

Configuring LDAP Authentication in Windows

You can configure the connector to use the LDAP protocol to authenticate the connection.

Note: If LDAP authentication is enabled, then SSL is automatically enabled.

To configure LDAP authentication in Windows:

1. Open the ODBC Data Source Administrator where you created the DSN, select the DSN, and then click **Configure**.
2. From the **Authentication Type** drop-down list, select **LDAP Authentication**.
3. In the **User** field, type an appropriate user name for accessing the data store.
4. In the **Password** field, type the password corresponding to the user name that you specified above.
5. To encrypt your credentials, select one of the following:
 - If the credentials are used only by the current Windows user, select **Current User Only**.
 - Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.
6. To configure client-server verification over SSL, click **SSL Options**. For more information, see [Configuring SSL Verification in Windows](#).
7. To save your settings and close the dialog box, click **OK**.

You can now use the connector to authenticate through LDAP and connect to your Trino server.

Configuring OIDC Authentication in Windows

You can configure the connector to use the OpenID Connect (OIDC) protocol to authenticate the connection.

To configure OIDC authentication in Windows:

1. Open the ODBC Data Source Administrator where you created the DSN, select the DSN, and then click **Configure**.
2. From the **Authentication Type** drop-down list, select **OIDC Authentication**.
3. In the **User** field, type the user name for accessing the data store.
4. To configure client-server verification over SSL, click **SSL Options**. For more information, see [Configuring SSL Verification in Windows](#).
5. To save your settings and close the dialog box, click **OK**.

6. When you test the connection or when you connect to your Trino server, a browser opens to the redirect URI. Type your credentials and click OK.

Configuring JWT Authentication in Windows

You can configure the connector to use the JSON Web Token (JWT) protocol to authenticate the connection.

To configure JWT authentication in Windows:

1. Open the ODBC Data Source Administrator where you created the DSN, select the DSN, and then click **Configure**.
2. From the **Authentication Type** drop-down list, select **JWT Authentication**.
3. In the **User** field, type the user name for accessing the data store.
4. In the **Access Token** field, type the access token corresponding to the user name that you specified above.
5. Click **Token Options**. The Access Token Options dialog box opens.
6. In the Access Token Options dialog box, select the desired option.
7. To configure client-server verification over SSL, click **SSL Options**. For more information, see [Configuring SSL Verification in Windows](#).
8. To save your settings and close the dialog box, click **OK**.

You can now use the connector to authenticate through JWT and connect to your Trino server.

Configuring AzureAD Authentication in Windows

You can configure the connector to use the AzureAD protocol to authenticate the connection.

To configure AzureAD authentication in Windows:

1. Open the ODBC Data Source Administrator where you created the DSN, select the DSN, and then click **Configure**.
2. From the **Authentication Type** drop-down list, select **AzureAD Authentication**.
3. In the **Client ID** field, type the client ID that is assigned to your application.
4. In the **Client Secret** field, type the client secret value generated in the application registration portal.
5. In the **Tenant** field, enter the granted permission for your application that is requested in GUID or domain name format.
6. In the **Scope** field, type the resource identifier affixed with the .default suffix value.
7. Optionally, in the **Authority** field, type `login.microsoftonline.com`.
8. Optionally, in the **refresh_token_interval** field, type the value in seconds.
9. To configure client-server verification over SSL, click **SSL Options**. For more information, see [Configuring SSL Verification in Windows](#).
10. To save your settings and close the dialog box, click **OK**.

You can now use the connector to authenticate through AzureAD and connect to your Trino server.

Configuring SSL Verification in Windows

If you are connecting to a Trino server that has Secure Sockets Layer (SSL) enabled, you can configure the connector to connect to an SSL-enabled socket. When using SSL to connect to a server, the connector can be configured to verify the identity of the server.


Note: If Kerberos or LDAP authentication is enabled, then SSL is automatically enabled.

To configure SSL verification in Windows:

1. To access SSL options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **SSL Options**.
 2. Select the **Enable SSL** check box.
 3. To allow authentication using self-signed certificates that have not been added to the list of trusted certificates, select the **Allow Self-signed Server Certificate** check box.
 4. To allow the common name of a CA-issued SSL certificate to not match the host name of the Trino server, select the **Allow Common Name Host Name Mismatch** check box.
 5. To specify the CA certificates that you want to use to verify the server, do one of the following:
 - To verify the server using the trusted CA certificates from a specific .pem file, specify the full path to the file in the **Trusted Certificates** field and clear the **Use System Trust Store** check box.
 - Or, to use the trusted CA certificates .pem file that is installed with the connector, leave the default value in the **Trusted Certificates** field, and clear the **Use System Trust Store** check box.
 - Or, to use the Windows trust store, select the **Use System Trust Store** check box.
- Important:**

 - If you are using the Windows trust store, make sure to import the trusted CA certificates into the trust store.
 - If the trusted CA supports certificate revocation, select the **Check Certificate Revocation** check box.
6. From the **Minimum TLS Version** drop-down list, select the minimum version of TLS to use when connecting to your data store.
 7. To configure two-way SSL verification, select the **Two-Way SSL** check box and then do the following:
 - a. In the **Client Certificate File** field, specify the full path of the PEM file containing the client's certificate.
 - b. In the **Client Private Key File** field, specify the full path of the file containing the client's private key.

- c. If the private key file is protected with a password, type the password in the **Client Private Key Password** field.

 **Important:** The password is obscured, that is, not saved in plain text. However, it is still possible for the encrypted password to be copied and used.

- d. To encrypt your credentials, select one of the following:
 - If the credentials are used only by the current Windows user, select **Current User Only**.
 - Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.

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

Configuring a Proxy Connection in Windows


If you are connecting to the data source through a proxy server, you must provide connection information for the proxy server.

To configure a proxy server connection in Windows:

1. To access proxy server options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Proxy Options**.
2. Select the **Use Proxy Server** check box.
3. In the **Proxy Host** field, type the host name or IP address of the proxy server.
4. In the **Proxy Port** field, type the number of the TCP port that the proxy server uses to listen for client connections.
5. Optionally, to save the proxy server password in the Windows registry, select **Save Password (Encrypted)**.
6. To save your settings and close the Proxy Options dialog box, click **OK**.

Configuring FIPS in Windows

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

 **Note:** To enable FIPS support, a separate FIPS downloadable package is available for assistance.

To configure FIPS in Windows:

1. Unzip the OpenSSL_3.0_Modules_Windows_vs2022.zip Windows package released with the connector.

2. Follow the instructions mentioned in the **README.md** file.



Note: Make sure that all the FIPS module binary files are present in the OPENSSL_MODULES path, otherwise the connector does not work as expected.

Exporting a Data Source Name in Windows

After you configure a DSN, you can export it to be used on other machines. When you export a DSN, all of its configuration settings are saved in a `.sdc` file. You can then distribute the `.sdc` file to other users so that they can import your DSN configuration and use it on their machines.

To export a Data Source Name in Windows:

1. Open the ODBC Data Source Administrator where you created the DSN, select the DSN, click **Configure**, and then click **Logging Options**.
2. Click **Export Configuration**, specify a name and location for the exported DSN, and then click **Save**.

Your DSN is saved as a `.sdc` file in the location that you specified.

Importing a Data Source Name in Windows

You can import a DSN configuration from a `.sdc` file and then use those settings to connect to your data source.

To import a Data Source Name in Windows:

1. Open the ODBC Data Source Administrator where you created the DSN, select the DSN, click **Configure**, and then click **Logging Options**.
2. Click **Import Configuration**, browse to select the `.sdc` file that you want to import the DSN configuration from, and then click **Open**.
3. Click **OK** to close the Logging Options dialog box.

The SimbaTrino ODBC Driver DSN Setup dialog box loads the configuration settings from the selected `.sdc` file. You can now save this DSN and use it to connect to your data source.

Configuring Logging Options in Windows

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



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

Configuring Connector-wide Logging Options

The settings for logging apply to every connection that uses the Simba Trino ODBC Connector, so make sure to disable the feature after you are done using it. To configure logging for the current connection, see [Configuring Logging for the Current Connection](#).

To enable connector-wide logging in Windows:

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

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

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

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

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

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

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

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

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

To disable connector logging in Windows:

1. Open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Logging Options**.
2. From the **Log Level** drop-down list, select **LOG_OFF**.

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

Configuring Logging for the Current Connection

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

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

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

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

To add logging configurations to a DSN in Windows:

1. Navigate to the appropriate registry key for the bitness of your connector and your machine:
 - 32-bit System DSNs: **HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\ODBC\ODBC.INI***[DSN Name]*
 - 64-bit System DSNs: **HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI***[DSN Name]*
 - 32-bit and 64-bit User DSNs: **HKEY_CURRENT_USER\SOFTWARE\ODBC\ODBC.INI***[DSN Name]*
2. For each configuration option that you want to configure for the current connection, create a value by doing the following:
 - a. If the key name value does not already exist, create it. Right-click the *[DSN Name]* and then select **New > String Value**, type the key name of the configuration option, and then press **Enter**.
 - b. Right-click the key name and then click **Modify**.
To confirm the key names for each configuration option, [Connector Configuration Options](#).
 - c. In the Edit String dialog box, in the **Value Data** field, type the value for the configuration option.
3. Close the Registry Editor.
4. Restart your ODBC application to make sure that the new settings take effect.

Verifying the Connector Version Number in Windows

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

To verify the connector version number in Windows:

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



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

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

macOS Connector

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

macOS System Requirements

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

- 150MB of available disk space
- One of the following ODBC driver managers installed:
 - iODBC 3.52.9 or later
 - unixODBC 2.3.6 or later

Installing the Connector in macOS

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

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

To install the Simba Trino ODBC Connector in macOS:

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



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

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

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

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

Verifying the Connector Version Number in macOS

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

To verify the connector version number in macOS:

- At the Terminal, run the command:

```
pkgutil --info com.simba.trinojdbc
```

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

Configuring FIPS in macOS

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

Note: To enable FIPS support, a separate FIPS downloadable package is available for assistance.

To configure FIPS in macOS:

- Unzip the `OpenSSL_3.0_Modules_OSX_ARM_xcode13_2.tar.gz` package released with the connector.
- Follow the instructions mentioned in the **README.md** file.

Note: Ensure that all the FIPS module binary files are present in the `OPENSSL_MODULES` path, otherwise the connector does not work as expected.

Uninstalling the Connector in macOS

You can uninstall the Simba Trino ODBC Connector in macOS by deleting the added files and folders from the Library.

To uninstall the Simba Trino ODBC Connector in macOS:

1. In the Finder, navigate to the location where the connector was installed.



Note: By default, the driver files are installed in the `/Library/simba/trinoodbc` directory.

2. Move all files and folders in this location to the Trash.

Linux Connector

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

Linux System Requirements

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

- 90MB of available disk space
- One of the following ODBC driver managers installed:
 - iODBC 3.52.9 or later
 - unixODBC 2.3.6 or later
- The `krb5-libs` library that matches the bitness of the connector must be installed.



Note: If the package manager in your Linux distribution cannot resolve the dependency automatically when installing the connector, then download and manually install the package.

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

Installing the Connector Using the RPM File

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

The placeholders in the file names are defined as follows:

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

You can install both the 32-bit and 64-bit versions of the connector on the same machine. For ARM machines, we support only 64-bit versions.

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

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

- If you are using Red Hat Enterprise Linux or CentOS, run the following command:

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

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

```
zypper install [RPMFileName]
```

- Or, if you are using Linux ARM Server, navigate to the `Linux_ARM` folder containing the RPM and run the following command:

```
Linux_ARM/zypper install [RPMFileName]
```

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

Installing the Connector on Debian

To install the connector on a Debian machine, use the Debian package instead of the RPM file.

On 64-bit editions of Debian(Non-ARM machine), you can execute both 32- and 64-bit applications. In this case, 64-bit applications must use 64-bit connectors, and 32-bit applications must use 32-bit connectors. However, on 64-bit editions of Debian(ARM architecture), we only provide 64-bit drivers. Make sure that you use the version of the connector that matches the bitness of the client application:

- `trino_[Version]-[Release]_i386.deb` for the 32-bit connector
- `trino_[Version]-[Release]_amd64.deb` for the 64-bit connector
- `trino_[Version]-[Release]_aarch64.deb` for the ARM 64-bit driver.

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

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

Note: For Linux ARM, only the 64-bit version of the connector is supported.

To install the Simba Trino ODBC Connector on Debian:

1. Log in as the root user, and then navigate to the folder containing the Debian package for the connector.
2. Double-click `trino_[Version]-[Release]_i386.deb` or `trino_[Version]-[Release]_amd64.deb` or `trino_[Version]-[Release]_aarch64.deb`.

3. Follow the instructions in the installer to complete the installation process.

The Simba Trino ODBC Connector files are installed in the `/opt/simba/trinojdbc` directory.

4. If you received a license file via email, then copy the license file into the `/opt/simba/trinojdbc/lib/32` or `/opt/simba/trinojdbc/lib/64` folder, depending on the version of the connector that you installed. You must have root privileges when changing the contents of this folder.

To install the Simba Trino ODBC Connector on Debian (Linux ARM):

1. Log in as the root user, and then navigate to the `Linux_ARM` folder containing the Debian package for the connector.
2. Double-click `simbatrinoodbc_[Version]-[Release]_amd64.deb`.
3. Follow the instructions in the installer to complete the installation process. The Simba Trino ODBC Connector files are installed in the `/opt/simbatrinoodbc` directory.
4. If you received a license file via email, then copy the license file into the `/opt/simba/trinojdbc/64` folder. You must have root privileges when changing the contents of this folder.

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

Verifying the Connector Version Number in Linux

If you need to verify the version of the Simba Trino ODBC Connector that is installed on your Linux machine, you can query the version number through the command-line interface if the connector was installed using an RPM file. Alternatively, you can search the connector's binary file for version number information.

To verify the connector version number in Linux using the command-line interface:

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

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

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

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

Configuring FIPS in Linux

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

Note: To enable FIPS support, a separate FIPS downloadable package is available for assistance.

To configure FIPS in Linux:

1. Unzip the OpenSSL_3.0_Modules_Linux_gcc5_5.tar.gz Linux package released with the connector.
2. Follow the instructions mentioned in the **README.md** file.

Note: Make sure that all the FIPS module binary files are present in the OPENSSL_MODULES path, otherwise the connector does not work as expected.

Configuring the ODBC Driver Manager in Non-Windows Machines

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

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

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

Specifying ODBC Driver Managers in Non-Windows Machines

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

macOS

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

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

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

Linux

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

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

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

Specifying the Locations of the Connector Configuration Files

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

If you are using iODBC, do the following:

- Set ODBCINI to the full path and file name of the `odbc.ini` file.
- Set ODBCINSTINI to the full path and file name of the `odbcinst.ini` file.
- Set SIMBATRINOINI to the full path and file name of the `simba.trinoodbc.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 ODBCYSINI to the full path of the directory that contains the `odbcinst.ini` file.
- Set SIMBATRINOINI to the full path and file name of the `simba.trinoodbc.ini` file.

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

For unixODBC:

```
export ODBCINI=/usr/local/odbc/odbc.ini
export ODBCYSINI=/usr/local/odbc
export SIMBATRINOINI=/etc/simba.trinoodbc.ini
```

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

1. If the SIMBATRINOINI environment variable is defined, then the connector searches for the file specified by the environment variable.
2. The connector searches the directory that contains the connector library files for a file named `simba.trinoodbc.ini`.
3. The connector searches the current working directory of the application for a file named `simba.trinoodbc.ini`.

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

Configuring ODBC Connections in Non-Windows Machine

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

- [Creating a Data Source Name in a Non-Windows Machine](#)
- [Configuring a DSN-less Connection in a Non-Windows Machine](#)
- [Configuring Authentication in a Non-Windows Machine](#)
- [Configuring SSL Verification in a Non-Windows Machine](#)
- [Configuring Logging Options in a Non-Windows Machine](#)
- [Testing the Connection in Non-Windows Machine](#)

Creating a Data Source Name in a Non-Windows Machine

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

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

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

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



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

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

For example, on a macOS machine:

```
[ODBC Data Sources]
```

```
Sample DSN=Simba Trino ODBC Connector
```

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

```
[ODBC Data Sources]
```

```
Sample DSN=Simba Trino ODBC Connector 32-bit
```

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

For example, on a macOS machine:

`Driver=/Library/simba/trinoodbc/lib/libtrinoodbc_sbu.dylib`

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

`Driver=/opt/simba/trinoodbc/lib/32/libtrinoodbc_sb32.so`

- b. Set the `Host` property to the IP address or host name of the server, and then set the `Port` property to the number of the TCP port that the server uses to listen for client connections.

For example:

`Host=192.168.222.160`

`Port=8080`

- c. If authentication is required to access the server, then specify the authentication mechanism and your credentials. For more information, see [Configuring Authentication in a Non-Windows Machine](#).
 - d. Optionally, set additional key-value pairs as needed to specify other optional connection settings. For detailed information about all the configuration options supported by the Simba Trino ODBC Connector, see [Connector Configuration Options](#).

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



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

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

[ODBC Data Sources]

Sample DSN=Simba Trino ODBC Connector

[Sample DSN]

`Driver=/Library/simba/trinoodbc/lib/libtrinoodbc_sbu.dylib`

`Host=192.168.222.160`

`Port=8080`

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

[ODBC Data Sources]

Sample DSN=Simba Trino ODBC Connector 32-bit

[Sample DSN]

Driver=/opt/simba/trinoodbc/lib/32/libtrinoodbc_sb32.so

Host=192.168.222.160

Port=8080

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

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

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

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

To define a connector on a non-Windows machine:

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



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

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

For example:

[ODBC Drivers]

Simba Trino ODBC Connector=Installed

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

For example, on a macOS machine:

Driver=/Library/simba/trinoodbc/lib/libtrinoodbc_sbu.dylib

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

Driver=/opt/simba/trinoodbc/lib/32/libtrinoodbc_sb32.so

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

For example:

Description=Simba Trino ODBC Connector

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



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

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

[ODBC Drivers]

Simba Trino ODBC Connector=Installed

[Simba Trino ODBC Connector]

Description=Simba Trino ODBC Connector

Driver=/Library/simba/trinoodbc/lib/libtrinoodbc_sbu.dylib

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

[ODBC Drivers]

Simba Trino ODBC Connector 32-bit=Installed

Simba Trino ODBC Connector 64-bit=Installed

[Simba Trino ODBC Connector 32-bit]

Description=Simba Trino ODBC Connector (32-bit)

Driver=/opt/simba/trinoodbc/lib/32/libtrinoodbc_sb32.so

[Simba Trino ODBC Connector 64-bit]

Description=Simba Trino ODBC Connector (64-bit)

Driver=/opt/simba/trinoodbc/lib/64/libtrinoodbc_sb64.so

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

Configuring Authentication in a Non-Windows Machine

Some Trino data stores require authentication. You can configure the Simba Trino ODBC Connector to provide your credentials and authenticate the connection to the database using one of the following methods:

- [Configuring Kerberos Authentication in a Non-Windows Machine](#)
- [Configuring LDAP Authentication in a Non-Windows Machine](#)

- [Configuring JWT Authentication in a Non-Windows Machine](#)

Note: If Kerberos or LDAP authentication is enabled, then SSL is automatically enabled.

Configuring Kerberos Authentication in a Non-Windows Machine

You can configure the connector to use the Kerberos protocol to authenticate the connection. You can set the connection properties 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.

Kerberos must be installed and configured before you can use this authentication mechanism. For information about how to install and configure Kerberos, see the MIT Kerberos Documentation: <http://web.mit.edu/kerberos/krb5-latest/doc/>.

When you configure your Kerberos server, in the `/etc/trino/config.properties` file, set the following properties:

- `http.server.authentication.krb5.service-name=HTTP`
- `http.server.authentication.krb5.keytab=HTTP.keytab`

When you use Kerberos authentication, the connector loads the credentials from the Kerberos credential cache. Therefore, a Kerberos ticket must be generated before you run the connector. To generate a Kerberos ticket, run the `kinit` Kerberos command with the appropriate principal.

Additionally, when using Kerberos authentication:

- The connector sends the Kerberos default user principal name as the user name.
- When the Kerberos ticket is generated, the default user principal name is retrieved from the Kerberos credential cache.
- Or, if you would like to manually send a user name, pass it via the `UID` connection parameter in the connection string.

Note: If Kerberos authentication is enabled, then SSL is automatically enabled.

To configure the connector to use Kerberos authentication on a non-Windows machine:

1. Run the `kinit` command, using the following syntax, where `[Principal]` is the Kerberos user principal to use for authentication:

```
kinit -k [Principal]
```
2. In your `odbc.ini` configuration file or connection string, set the `AuthenticationType` property to `Kerberos Authentication`.
3. Optionally, to generate a ticket using a Kerberos user name and password:
 - a. Set the `UseExistingKrbCreds` property to `1` use the existing Kerberos Credentials, or to `0` to generate new credentials.
 - b. Set the `KinitType` property to `Kinit with Password`.

- c. Optionally, to forward the generated Kerberos credentials, set the `DelegateKrbCreds` property to 1.
 - d. Set the `KerberosUsername` property to your Kerberos user name.
 - e. Set the `KerberosPassword` property to your Kerberos password.
4. Optionally, to generate a ticket using a Kerberos user name and a keytab file:
 - a. Set the `UseExistingKrbCreds` property to 1 use the existing Kerberos Credentials, or to 0 to generate new credentials.
 - b. Set the `KinitType` property to `Kinit with Keytab`.
 - c. Optionally, to forward the generated Kerberos credentials, set the `DelegateKrbCreds` property to 1.
 - d. Set the `KerberosUsername` property to your Kerberos user name.
 - e. Set the `KerberosKeytab` property to the full path of the keytab file.
5. Optionally, to use a service name other than the default of HTTP, set the `KrbServiceName` property to the service name of the Trino server.
6. Configure the SSL settings as described in [Configuring SSL Verification in a Non-Windows Machine](#).

You can now use the connector to authenticate through Kerberos and connect to your Trino server.

Configuring LDAP Authentication in a Non-Windows Machine

You can configure the connector to use the LDAP protocol to authenticate the connection. You can set the connection properties in a connection string or in a DSN (in the `odbc.ini` file). Settings in the connection string take precedence over settings in the DSN.

Note: If LDAP authentication is enabled, then SSL is automatically enabled.

To configure LDAP authentication on a non-Windows machine:

1. Set the `AuthenticationType` property to `LDAP Authentication`.
2. Set the `UID` property to an appropriate user name for accessing the data store.
3. Set the `PWD` property to the password corresponding to the user name that you specified above.
4. Configure the SSL settings as described in [Configuring SSL Verification in a Non-Windows Machine](#).

You can now use the connector to authenticate through LDAP and connect to your Trino server.

Configuring OIDC Authentication in a Non-Windows Machine

You can configure the connector to use the OpenID Connect (OIDC) protocol to authenticate the connection. You can set the connection properties 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.

To configure OIDC authentication on a non-Windows machine:

1. Set the `AuthenticationType` property to `OIDC Authentication`.
2. Set the `UID` property to an appropriate user name for accessing the data store.
3. Configure the SSL settings as described in [Configuring SSL Verification in a Non-Windows Machine](#).
4. When you test the connection or when you connect to your Trino server, a browser opens to the redirect URI. Type your credentials and click **OK**.

The connector retrieves an access token, and you can connect to your Trino server. If the access token expires or become invalid, it will be automatically renewed.

Configuring JWT Authentication in a Non-Windows Machine

You can configure the connector to use the JSON Web Token (JWT) protocol to authenticate the connection. You can set the connection properties 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.

To configure JWT authentication on a non-Windows machine:

1. Set the `AuthenticationType` property to `JWT Authentication`.
2. Set the `UID` property to an appropriate user name for accessing the data store.
3. Set the `AccessToken` property to the access token corresponding to the user name that you specified above.
4. Configure the SSL settings as described in [Configuring SSL Verification in a Non-Windows Machine](#).

You can now use the connector to authenticate through JWT and connect to your Trino server.

Configuring SSL Verification in a Non-Windows Machine

If you are connecting to a Trino server that has Secure Sockets Layer (SSL) enabled, you can configure the connector to connect to an SSL-enabled socket. When using SSL to connect to a server, the connector can be configured to verify the identity of the server.



Note: If either Kerberos or LDAP authentication are enabled, the connector automatically uses SSL to communicate with the Trino server.

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

To configure SSL verification on a non-Windows machine:

1. To enable SSL connections, set the `SSL` attribute to `1`.
2. To allow authentication using self-signed certificates that have not been added to the list of trusted certificates, set the `AllowSelfSignedServerCert` attribute to `1`.

3. To allow the common name of a CA-issued SSL certificate to not match the host name of the Trino server, set the `AllowHostNameCNMismatch` attribute to 1.
4. Choose one:
 - To configure the connector to load SSL certificates from a specific `.pem` file when verifying the server, set the `TrustedCerts` attribute to the full path of the `.pem` file.
 - Or, to use the trusted CA certificates `.pem` file that is installed with the connector, do not specify a value for the `TrustedCerts` attribute.
5. To configure two-way SSL verification, set the `TwoWaySSL` attribute to 1 and then do the following:
 - a. Set the `ClientCert` attribute to the full path of the `.pem` file containing the client's certificate.
 - b. Set the `ClientPrivateKey` attribute to the full path of the file containing the client's private key.
 - c. If the private key file is protected with a password, set the `ClientPrivateKeyPassword` attribute to the password.
6. To allow authentication, when the certificate's revocation status is undetermined, set the `Accept Undetermined Revocation` attribute to 1.
7. To specify the minimum version of TLS to use, set the `Min_TLS` property to the minimum version of TLS. Supported options include 1.0 for TLS 1.0, 1.1 for TLS 1.1, 1.2 for TLS 1.2, and 1.3 for TLS 1.3.

Configuring Logging Options in a Non-Windows Machine

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



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

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

To enable logging on a non-Windows machine:

1. To specify the level of information to include in log files, set the `LogLevel` property to one of the following numbers:

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

LogLevel Value	Description
5	Logs detailed information that is useful for debugging the connector.
6	Logs all connector activity.

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

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

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

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

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

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

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

To disable logging on a non-Windows machine:

1. Set the `LogLevel` key to 0.
2. Save the `simba.trinojdbc.ini` configuration file.
3. Restart your ODBC application to make sure that the new settings take effect.

Testing the Connection in Non-Windows Machine

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

Using the iODBC Driver Manager

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

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

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

To test your connection using the iODBC driver manager:

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

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

Using the unixODBC Driver Manager

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

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

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

To test your connection using the unixODBC driver manager:

- Run `isql` or `iusql` by using the corresponding syntax:

```
▪ isql [DataSourceName]
▪ iusql [DataSourceName]
```

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

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

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

Using a Connection String

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

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

DSN Connection String Example

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

`DSN=[DataSourceName]`

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

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

DSN-less Connection String Examples

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

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

- `[AuthType]` is the method that is used for authentication.
- `[PortNumber]` is the number of the port that the Trino server uses to listen for client connections.
- `[Server]` is the IP address or host name of the Trino server to which you are connecting.
- `[YourAccessToken]` is the access token that you use to access the Trino server.
- `[YourUserName]` is the user name that you use to access the Trino server.

The following is the format of a DSN-less connection string:

`Driver=Simba Trino ODBC Driver;`
`Host=[Server];Port=[PortNumber];AuthMech=[Auth_Type];`

For example:

`Driver=Simba Trino ODBC Driver;`
`Host=http://host%3Dorg.api.imply.io/;Port=443;AuthMech=0`

Connecting to Trino Using Basic Authentication

The following is the format of a DSN-less connection string for connecting to Trino using basic authentication:

```
Driver=SimbaTrino ODBC Driver;Host=[YourInstanceURL];Auth_Type=Basic;UID=[YourAPIKey]
```

For example:

```
Driver=SimbaTrino ODBC Driver;Host=https://api.stripe.com;Auth_Type=Basic;UID=sk_test_
cXg9KLwthuyyu980694
```

Connecting to Trino Using Basic Auth (API Key authentication)

You can connect to the Trino server by using API Key authentication. The following is the format of a DSN-less connection string for connecting to Trino using API Key authentication:

```
Driver= Simba Trino ODBC Driver;
```

```
Host=[Server];Port=[PortNumber];AuthMech=[Auth_Type];UID=[Your_API_Key];PWD=
[YourPassword]
```

For example:

```
Driver= Simba Trino ODBC Driver;
```

```
Host=http://host%3Dorg.api.imply.io;Port=443;AuthMech=0;UID=some_API_Key;
```

Features

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

- [Catalog and Schema Support](#)
- [Parameters](#)
- [Resource Group](#)
- [Data Types](#)
- [Security and Authentication](#)

Catalog and Schema Support

The Simba Trino ODBC Connector supports both catalogs and schemas to make it easy for the connector to work with various ODBC applications.

Note: SQLTables and SQLColumns only work if a catalog has been specified by the user in the connection string or set as a connection attribute.

The Simba Trino ODBC Connector supports querying against Hive, MySQL, and PostgreSQL schemas.

Parameters

A parameterized query contains placeholders that are used for parameters. The values of those parameters are supplied at execution time.

The Simba Trino ODBC Connector fully supports parameterized queries. If Auto Populate Parameter Metadata is selected or the `AutoIPD` option is set to 1, the connector automatically populates the metadata for parameters.

Resource Group

Resource groups are a Trino feature that allows administrators to control resource usage and query scheduling.

To use this feature, define either of the following properties:

- Application Name (or `ApplicationName`)
- ClientTags (or `ClientTags`)
- Application Name Prefix (or `ApplicationNamePrefix`) if required.

If the Trino server has a resource group that selects for those values, then the queries are executed according to the policies defined for that resource group.

Data Types

The Simba Trino ODBC Connector supports many common SQL data types.

The table below lists the supported data types.

Supported SQL types	
ARRAY <i>i</i> Note: The connector casts this type to VARCHAR.	REAL <i>i</i> Note: Only supported in Trino 0.152t and later.
BIGINT	ROW <i>i</i> Note: The connector casts this type to VARCHAR.
BOOLEAN	SMALLINT
CHAR(x) <i>i</i> Note: WCHAR is used instead if the Use Unicode SQL Character Types configuration option (the UseUnicodeSqlCharacterTypes key) is enabled.	TIME
Supported SQL types <i>i</i> Note: For all VARCHAR types, WVARCHAR is used instead if the Use Unicode SQL Character Types configuration option (the UseUnicodeSqlCharacterTypes key) is enabled.	TIME(P)
DATE	TIME WITH TIME ZONE
VARCHAR	TIME(P) WITH TIME ZONE
DECIMAL	TIMESTAMP
DECIMAL	TIMESTAMP(P)
DOUBLE	TIMESTAMP WITH TIME ZONE
INTEGER	VARBINARY
INTERVAL DAY TO SECOND	VARCHAR (fixed length)

Supported SQL types	
INTERVAL YEAR TO MONTH	VARCHAR (variable length)
FLOAT Note: Deprecated in Trino 0.152t and later.	TINYINT
MAP Note: The connector casts this type to VARCHAR.	VARCHAR(x)

Security and Authentication

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

Note: In this documentation, "SSL" refers to both TLS (Transport Layer Security) and SSL (Secure Sockets Layer). The connector supports TLS 1.0, 1.1, and 1.2. The SSL version used for the connection is the highest version that is supported by both the connector and the server.

The connector provides a mechanism that enables you to authenticate your connection using the Kerberos protocol or the LDAP protocol. For detailed configuration instructions, see [Configuring Authentication in Windows](#) or [Configuring Authentication in a Non-Windows Machine](#).

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

- One-way authentication
- Two-way authentication

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

Connector Configuration Options

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

When creating or configuring a connection from a Windows computer, the fields and buttons are available in the Simba Trino ODBC Driver DSN Setup dialog box. When using a connection string or configuring a connection from a Linux or macOS computer, use the key names provided.

Configuration Options Appearing in the User Interface

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

<ul style="list-style-type: none"> ▪ Allow Common Name Host Name Mismatch ▪ Allow HTTP Redirect ▪ Allow Metadata From Multiple Catalogs ▪ Allow Self-Signed Server Certificate ▪ ApplicationName ▪ Application Name Prefix ▪ Assume Literal Underscore In Metadata Calls ▪ Authority ▪ AutoCommit Always True ▪ Auto Populate Parameter Metadata ▪ Catalog ▪ Client ID ▪ Client Certificate File ▪ Client Private Key File ▪ Client Private Key Password ▪ Client Secret ▪ ClientTags ▪ Connection Test 	<ul style="list-style-type: none"> ▪ MaxCatalogNameLen ▪ Max Column Name Length ▪ Max Complex Type Column Length ▪ Max File Size ▪ Max Number Files ▪ Max Prepared Statement Length ▪ Max Schema Name Length ▪ Max Table Name Length ▪ Max Varchar Column Length ▪ Minimum TLS ▪ Mute Server Warnings ▪ Password ▪ Port ▪ Proxy Host ▪ Proxy Password ▪ Proxy Port ▪ Proxy Uid ▪ Schema ▪ Scope ▪ Server Version ▪ Service Name
---	--

<ul style="list-style-type: none"> ▪ Delegate Kerberos Credentials ▪ Enable SSL ▪ Encrypt Password ▪ ExtraCredentials ▪ Host ▪ Kerberos Password ▪ Kerberos Username ▪ Keytab File Path ▪ Kinit Type ▪ Log Level ▪ Log Path 	<ul style="list-style-type: none"> ▪ Tenant Tenant ▪ Time Zone ID ▪ Trusted Certificates ▪ Two-Way SSL ▪ Use DSN Schema For Metadata ▪ Use Equal In Metadata Filters ▪ Use Existing Kerberos Credentials ▪ Use GSSAPI ▪ Use Proxy Server ▪ Use System Catalog For Metadata ▪ Use System Trust Store ▪ Use Unicode SQL Character Types ▪ User
--	---

When creating or configuring a connection from a Windows computer, the fields and buttons are available in the Trino ODBC Driver DSN Setup dialog box. When using a connection string or configuring a connection from a Linux or macOS computer, use the key names provided.

Accept Undetermined Revocation

This option specifies whether the connector allows an SSL connection when the certificate's revocation status is undetermined.

- Enabled(1): The connector allows an SSL connection even when the certificate's revocation status is undetermined.
- Disabled(0): The connector does not allow SSL connection when the certificate's revocation status is undetermined.

Key Name	Default Value	Required
AcceptUndeterminedRevocation	0	No

Access Token Cache Location

This option specifies the location of all the cached access token files.

Key Name	Default Value	Required
AccessTokenCacheLocation	None	No

Note: If the location is undefined, then the files are stored in a temporary location.

Allow Common Name Host Name Mismatch

This option specifies whether a CA-issued SSL certificate name must match the host name of the Trino server.

Note: The key for this option used to be `CAIssuedCertNamesMismatch`, and is still recognized by the connector under that key. If both keys are defined, `AllowHostNameCNMismatch` will take precedence.

This setting is applicable only when SSL is enabled.

- Enabled (1): The connector allows a CA-issued SSL certificate name to not match the host name of the Trino server.
- Disabled (0): The CA-issued SSL certificate name must match the host name of the Trino server.

Key Name	Default Value	Required
AllowHostNameCNMismatch	Clear (0)	No

Allow HTTP Redirect

This options specifies whether the connector allows HTTP redirects.

- Enabled (1):The connector allows HTTP redirects.
- Disabled (0): The connector does not allow HTTP redirects.

Key Name	Default Value	Required
AllowHTTPRedirect	Disabled (0)	No

Allow Metadata From Multiple Catalogs

This option specifies whether metadata is retrieved from all catalogs when the connector makes a call to `SQLTables` or `SQLColumns`.

- Enabled (1): The connector retrieves metadata from all catalogs when making calls to `SQLTables` or `SQLColumns`, as per the ODBC standard.
- Disabled (0): The connector only retrieves metadata from the specified catalog when making calls to `SQLTables` or `SQLColumns`.


Note:

- If this option is disabled, you must specify a catalog to make calls to SQLTables or SQLColumns. You can specify a catalog in the call to SQLTables or SQLColumns, or in the Catalog DSN setting (the `Catalog` connection property).
- Disabling this option may improve connector performance.

Key Name	Default Value	Required
AllowMetadataFromMultipleCatalogs	Enabled (1)	No

Allow Self-Signed Server Certificate

This option specifies whether the connector allows a connection to a Trino server that uses a self-signed certificate, even if this certificate is not in the list of trusted certificates. This list is contained in the Trusted Certificates file, or in the system Trust Store if the system Trust Store is used instead of a file.

- Enabled (1): The connector authenticates the Trino server even if the server is using a self-signed certificate that has not been added to the list of trusted certificates.
- Disabled (0): The connector does not allow self-signed certificates from the server unless they have already been added to the list of trusted certificates.



Note: This setting is applicable only when SSL is enabled.

Key Name	Default Value	Required
AllowSelfSignedServerCert	Clear (0)	No

ApplicationName

Set this property to an application flag that you want to apply to the queries sent by the connector. If the application flag has been specified in a Trino resource group, then the queries are run according to the policies defined in that resource group.

Key Name	Default Value	Required
ApplicationName	None	No

Application Name Prefix

Use this property to apply any required prefixes to the Application Name (or `ApplicationName`) property.

Key Name	Default Value	Required
ApplicationNamePrefix	None	No

Assume Literal Underscore In Metadata Calls

This option specifies whether the connector matches underscore as a literal or not in the metadata queries.

- **Enabled (1):** The connector allows to match underscore as a literal in the metadata queries.
- **Disabled (0):** The connector does not escape underscore in the metadata queries.

Key Name	Default Value	Required
AssumeLiteralUnderscoreInMetadataCalls	Clear (0)	No

Authentication Type

This option specifies the type of authentication that the connector uses.

Select from the following:

- **No Authentication:** The connector does not authenticate the connection.
- **Kerberos Authentication:** The connector uses Kerberos to authenticate the connection. For more information about Kerberos authentication in Windows, see the Windows Kerberos documentation: [https://msdn.microsoft.com/en-us/library/windows/desktop/aa378747\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa378747(v=vs.85).aspx). For more information about Kerberos authentication in macOS or Linux, see the MIT Kerberos Documentation: <http://web.mit.edu/kerberos/krb5-latest/doc/>.
- **LDAP Authentication:** The connector uses LDAP to authenticate the connection.
- **JWT Authentication:** The connector uses a JSON Web Token (JWT) to authenticate the connection.



Note: If either Kerberos Authentication or LDAP Authentication are specified, the connector automatically uses SSL to communicate with the Trino server.

Key Name	Default Value	Required
AuthenticationType	No Authentication	No

Authority

This option specifies the authority of an Azure active directory end point.

Key Name	Default Value	Required
Authority	login.microsoftonline.com	No

AutoCommit Always True

This option specifies whether the connector always auto-commits.

- Enabled (1): The connector ignores the SQL_ATTR_AUTOCOMMIT value and auto-commits.
- Disabled (0): The connector considers the SQL_ATTR_AUTOCOMMIT value.

Key Name	Default Value	Required
AutoCommit	Disabled (0)	No

Auto Populate Parameter Metadata

This option specifies whether the connector automatically populates the parameter metadata for parameterized SQL statements.

The Trino server does not necessarily provide parameter metadata for every parameter in a parameterized SQL statement. When the server does not provide parameter metadata, the connector defines the parameter data type as SQL_VARCHAR.

Automatically populating parameter metadata may cause certain ODBC applications to not function correctly. In that case, this option should be disabled.

- Enabled (1): The connector automatically populates the metadata for parameters.
- Disabled (0): The connector does not automatically populate the metadata for parameters.

Key Name	Default Value	Required
AutoIPD	Enabled (1)	No

Cache Access Token

This options specifies whether the connector caches an access token.

- Enabled (1): The connector caches access tokens.
- Disabled (0): The connector does not cache access tokens.

Note: This setting is available for OIDC authentication only.

Key Name	Default Value	Required
cacheAccessToken	Disabled (0)	No

Calling App Name

This option passes the specified application name through the connector to use the application specific functionality.

- **PowerPivot:** When using Power Pivot in Excel, this options allows you to retrieve the list of tables and view table data using the "Select from a list of tables and views to choose the data to import" option.

Key Name	Default Value	Required
CallingAppName	NULL	No

Catalog

The current catalog context for all requests against the server.

Key Name	Default Value	Required
Catalog	None	No

CheckCertificate Revocation

This option specifies whether the connector checks to see if a certificate has been revoked while retrieving a certificate chain from the Windows Trust Store.

This option is only applicable if you are using a CA certificate from the Windows Trust Store (see [Use System Trust Store](#)).

- **Enabled (1):** The connector checks for certificate revocation while retrieving a certificate chain from the Windows Trust Store.
- **Disabled (0):** The connector does not check for certificate revocation while retrieving a certificate chain from the Windows Trust Store.

Note: This option is disabled when the `AllowSelfSignedServerCert` property is set to 1. This option is only available in Windows.

Key Name	Default Value	Required
CheckCertRevocation	Clear (0)	No

Client ID

This option specifies the client ID that is assigned to your Trino application.

Key Name	Default Value	Required
ClientID	None	Yes, if AuthenticationType= AAD

Key Name	Default Value	Required
		Client Secret authentication.

Client Certificate File

The full path to the .pem file containing the client's SSL certificate.

Note: This setting is applicable only when two-way SSL is enabled.

Key Name	Default Value	Required
ClientCert	None	No

Client Private Key File

The full path to the .pem file containing the client's SSL private key.

Note:
This setting is applicable only when two-way SSL is enabled.

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

Client Private Key Password

The password of the private key file that is specified in the Client Private Key File field (ClientPrivateKey).

Key Name	Default Value	Required
ClientPrivateKeyPassword	None	Yes, if two-way SSL verification is enabled and the client's private key file is protected with a password.

Client Secret

This option specifies the client secret value that is generated in the application registration portal.

Key Name	Default Value	Required
ClientSecret	None	Yes, if AuthenticationType= AAD Client Secret authentication.

ClientTags

Set this property to a comma-separated list of resource group tags that you want to apply to the queries sent by the connector. If the tags have been specified in a Trino resource group, then the queries are run according to the policies defined in that resource group.

Key Name	Default Value	Required
ClientTags	None	No

Connection Test

This option specifies whether the connector should automatically attempt to test the connection by contacting the server while establishing the connection.

- Enabled (1): The connector automatically tests the connection while establishing the connection.
- Disabled (0): The connector does not automatically test the connection.



Note:

- Disabling this option may improve connector performance.
- If this option is disabled, you should specify the version of the Trino server in the `Server Version` or `ServerVersion` configuration option (see [Server Version](#)).

Key Name	Default Value	Required
ConnectionTest	Enabled (1)	No

Delegate Kerberos Credentials

This options specifies whether the connector forwards the generated Kerberos credentials.

- Enabled (1): The connector forwards the generated Kerberos credentials.
- Disabled (0): The connector does not forward the generated Kerberos credentials.

Key Name	Default Value	Required
DelegateKrbCreds	Disabled (0)	No

Enable SSL

This option specifies whether the client uses an SSL encrypted connection to communicate with the Trino server.

- Enabled (1): The client communicates with the Trino server using SSL.
- Disabled (0): SSL is disabled.

SSL is configured independently of authentication. When authentication and SSL are both enabled, the connector performs the specified authentication method over an SSL connection.

Note: If either Kerberos Authentication or LDAP Authentication are specified, the connector automatically uses SSL to communicate with the Trino server.

Key Name	Default Value	Required
SSL	Clear (0)	No

Encrypt Password

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

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

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

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

ExtraCredentials

Set this property to a comma-separated list of key-value pairs that you want to pass to an external service.

For example, to set the key-value pairs `Hadoop=Trino` and `Driver=Trino`, you would set this property as follows:

`ExtraCredentials=Hadoop:Trino,Driver:Trino`

Key Name	Default Value	Required
ExtraCredentials	None	No

ExtraCredentialsDelimiter

Set this property to a one character delimiter to split up the key-value pairs that you want to pass to an external service.

For example, if `ExtraCredentialsDelimiter` is set to `+` and `ExtraCredentials` is set to the following:

```
sample_name+78,sample_fingerprint+191:156:116:8,sample_Key+value
```

This is the final output:

```
sample_name=78
```

```
sample_fingerprint=191:156:116:8
```

```
sample_Key=value
```

Key Name	Default Value	Required
<code>ExtraCredentialsDelimiter</code>	:	No

Force LowerCase UserName

This option makes sure that the username is in lowercase.

Key Name	Default Value	Required
<code>ForceLowerCaseUserName</code>	False	No

Host

The IP address or host name of the Trino server.

Key Name	Default Value	Required
<code>Host</code>	None	Yes

Keytab File Path

The full path to the keytab file used to generate Kerberos tickets.

Key Name	Default Value	Required
<code>KerberosKeytab</code>	None	Yes, if connecting using <code>kinit</code> with a Kerberos user name and a keytab file.

Kerberos Password

The Kerberos password used to generate tickets.

Key Name	Default Value	Required
KerberosPassword	None	Yes, if connecting using <code>kinit</code> with a Kerberos user name and password.

Kerberos Username

The Kerberos user name used to generate tickets.

Key Name	Default Value	Required
KerberosUsername	None	Yes, if connecting using <code>kinit</code> with a Kerberos user name and password or a keytab file.

Kinit Type

This option specifies whether the connector generates Kerberos tickets using `kinit` with a password or a keytab file.

- **Kinit with Password** (`Kinit with Password`): The connector generates tickets using a Kerberos user name and password.
- **Kinit with Keytab** (`Kinit with Keytab`): The connector generate tickets using a Kerberos user name and a keytab file.

Key Name	Default Value	Required
KinitType	None	Yes, if connecting using <code>kinit</code> with a Kerberos user name and password or a keytab file.

Log Level

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



Important:

- Only enable logging long enough to capture an issue. Logging decreases performance and can consume a large quantity of disk space.
- When logging with connection strings and DSNs, this option only applies to per-connection logs.

Set the property to one of the following values:

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

When logging is enabled, the connector produces the following log files at the location you specify in the Log Path (LogPath) property:

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

Key Name	Default Value	Required
LogLevel	OFF (0)	No

Log Path

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



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

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

MaxCatalogNameLen

This option specifies the maximum number of characters that the driver reports for SQL_MAX_CATALOG_NAME_LEN. This allows the applications to allocate a large enough buffer to retrieve the catalog name.

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

Key Name	Default Value	Required
MaxCatalogNameLen	0	No

Max Column Name Length

This option specifies the maximum number of characters that the driver reports for SQL_MAX_COLUMN_NAME_LEN. This allows the applications to allocate a large enough buffer to retrieve the column name.

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

Key Name	Default Value	Required
MaxColumnNameLen	0	No

Max Complex Type Column Length

The maximum data length for complex types that the connector casts to VARCHAR, that is, JSON, MAP, ROW, and ARRAY.

Key Name	Default Value	Required
MaxComplexTypeColumnLength	2048	No

Max File Size

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

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

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

Key Name	Default Value	Required
LogFileSize	20971520	No

Max Number Files

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

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

Key Name	Default Value	Required
LogFileCount	50	No

Max Prepared Statement Length

This option specifies the maximum length of the prepared statement.

Key Name	Default Value	Required
MaxPreparedStatementLength	7000	No

Max Schema Name Length

This option specifies the maximum number of characters that the driver reports for SQL_MAX_SCHEMA_NAME_LEN. This allows the applications to allocate a large enough buffer to retrieve the schema name.

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

Key Name	Default Value	Required
MaxSchemaNameLength	256	No

Max Table Name Length

This option specifies the maximum number of characters that the driver reports for SQL_MAX_TABLE_NAME_LEN. This allows the applications to allocate a large enough buffer to retrieve the table name.

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

Key Name	Default Value	Required
MaxTableNameLen	0	No

Max Varchar Column Length

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

Key Name	Default Value	Required
MaxDefaultVarCharLength	2048	No

Minimum TLS

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

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

Key Name	Default Value	Required
Min_TLS	TLS 1.3 (1.3)	No

Mute Server Warnings

This option specifies whether the driver disables the server warnings.

- Enabled (1): The driver does not allow server warnings.
- Disabled (0): The driver allows server warnings.

Key Name	Default Value	Required
MuteServerWarnings	Disabled (0)	No

Password

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



Note: This option is only available when the Authentication Type is set to LDAP Authentication.

Key Name	Default Value	Required
PWD	None	No

Port

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

Key Name	Default Value	Required
Port	8080	Yes

Proxy Host

The host name or IP address of a proxy server that you want to connect through.

Key Name	Default Value	Required
ProxyHost	None	Yes, if connecting through a

Key Name	Default Value	Required
		proxy server.

Proxy Password

The password that you use to access the proxy server.

Key Name	Default Value	Required
ProxyPwd	None	Yes, if connecting to a proxy server that requires authentication.

Proxy Port

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

Key Name	Default Value	Required
ProxyPort	None	Yes, if connecting through a proxy server.

Proxy Uid

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

Key Name	Default Value	Required
ProxyUid	None	Yes, if connecting to a proxy server that requires authentication.

Remove Type Name Parameters

This options specifies the removal of length, precision, or scale parameters from the SQLColumns() typename.

- Enabled (1): The driver removes length, precision, or scale parameters from the SQLColumns() typename.
- Disabled (0): The driver does not remove length, precision, or scale parameters from the SQLColumns() typename.

Key Name	Default Value	Required
RemoveTypeNameParameters	0	No

Roles

This option allows you to set roles for catalogs, as a list of key-value pairs for the catalog and role. The same value is provided for the X-Trino-Roles header field.

Key Name	Default Value	Required
Roles	None	No

Schema

The current schema context for all requests against the server.

Note: This option is only used when the `Catalog` option is specified and the value is not an empty string.

Key Name	Default Value	Required
Schema	None	No

Scope

This option specifies the resource identifier (application ID URI) of the resource you want.

Key Name	Default Value	Required
Scope	None	Yes, if AuthenticationType= AAD Client Secret authentication.

Server Version

This option specifies the version of the Trino server that the connector connects to, for example, `0.148-t`. This value is used when the connector cannot automatically detect the server version.

Note: If Connection Test is cleared or `ConnectionTest` is set to 0, this option should be set to the version of the Trino server that is being used.

Key Name	Default Value	Required
ServerVersion	None	No

Service Name

The Kerberos service principal name of the Trino server.

Key Name	Default Value	Required
KrbServiceName	HTTP	No

Skip Impersonation(Kerberos)

When the Skip Impersonation(Kerberos) is enabled, the x-trino-user header is not passed during Kerberos authentication if the user field is left empty.

Key Name	Default Value	Required
SkipImpersonationKerberos	False	No

Tenant

This option specifies the granted permissions requested in GUID or domain name format.

Key Name	Default Value	Required
Tenant	None	Yes, if AuthenticationType= AAD Client Secret authentication.

Time Zone ID

This option specifies the local time zone that the connector uses. Valid values for this option are specified in the IANA Time Zone Database. For a complete list of time zones, see https://en.wikipedia.org/wiki/List_of_tz_database_time_zones.

Key Name	Default Value	Required
TimeZoneID	System time zone	No

Trusted Certificates

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

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

Note: This setting is applicable only when SSL is enabled.

Key Name	Default Value	Required
TrustedCerts	The cacerts.pem file in the \lib subfolder within the connector's installation directory. The exact file path varies depending on the version of the	No

Key Name	Default Value	Required
	connector that is installed. For example, the path for the Windows connector is different from the path for the macOS connector.	

Two-Way SSL

This option specifies whether two-way SSL is enabled.

- Enabled (1): The client and the Trino server verify each other using SSL. See also the connector configuration options [Client Certificate File](#) and [Client Private Key File](#).
- Disabled (0): The server does not verify the client. Depending on whether one-way SSL is enabled, the client might verify the server. For more information, see [Enable SSL](#).



Note:

This option is applicable only when connecting to a Trino server that supports SSL. You must enable SSL before Two Way SSL can be configured. For more information, see [Enable SSL](#).

Key Name	Default Value	Required
TwoWaySSL	Clear (0)	No

Use DSN Schema For Metadata

When the schema is not specified, this option specifies whether the connector uses the schema name passed in the DSN for metadata queries. The schema passed takes precedence over the DSN schema.

- Enabled (true): The connector uses the schema name passed in the DSN for metadata queries.
- Disabled (false): The connector does not use the schema name passed in the DSN for metadata queries.

Key Name	Default Value	Required
UseDSNSchemaForMetadata	Clear (false)	No

Use Equal In Metadata Filters

This option specifies whether the connector uses an equal sign (=) or the `LIKE` keyword in metadata queries.

- Enabled (true): The connector uses an equal sign (=) in metadata queries.
- Disabled (false): The connector uses the `LIKE` keyword in metadata queries.

Key Name	Default Value	Required
UseEqualInMetadataFilters	Clear (false)	No

Use Existing Kerberos Credentials

This option specifies whether the connector uses existing Kerberos credentials or generates new Kerberos credentials.

- Enabled (1): The connector uses the existing Kerberos credentials.
- Disabled (0): The connector generates and uses new Kerberos credentials based on the `KinitType` settings.

Key Name	Default Value	Required
UseExistingKrbCreds	Enabled (1)	Yes, if connecting using <code>kinit</code> with a Kerberos user name and password or a keytab file.

Use GSSAPI

This option indicates whether the connector should use MIT Kerberos. To use this option, the MIT Kerberos library must be installed on the client machine. This option is only available in Windows.

- Enabled (1): The connector uses the MIT Kerberos library for Kerberos authentication.
- Disabled (0): The connector uses the Windows native SSP interface for Kerberos authentication.

Note: This option is only available in Windows.

Key Name	Default Value	Required
UseGSSAPI	Clear (0)	No

Use Proxy Server

This option specifies whether the connector uses a proxy server to connect to the data store.

- Enabled (1): The connector connects to a proxy server based on the information provided in the `Proxy Host`, `Proxy Port`, `Proxy Username`, and `Proxy Password` fields or the `ProxyHost`, `ProxyPort`, `ProxyUID`, and `ProxyPWD` keys.
- Disabled (0): The connector connects directly to the Trino server.

Key Name	Default Value	Required
UseProxy	Clear (0)	No

Use System Catalog For Metadata

This option specifies whether the connector uses the System Catalog or Information_Schema API to run metadata queries.

Key Name	Default Value	Required
UseSystemCatalogForMetadata		No

Use System Trust Store

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

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

Note: This option is only available in Windows.

Key Name	Default Value	Required
UseSystemTrustStore		No

Use Unicode SQL Character Types

This option specifies the SQL types to be returned for string data types.

- Enabled (1): The connector returns SQL_WVARCHAR for VARCHAR columns, and returns SQL_WCHAR for CHAR columns.
- Disabled (0): The connector returns SQL_VARCHAR for VARCHAR columns, and returns SQL_CHAR for CHAR columns.

Key Name	Default Value	Required
UseUnicodeSqlCharacterTypes	Selected (1)	No

User

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

Key Name	Default Value	Required
UID	TrinoODBC_Driver	No

Configuration Options Having Only Key Names

The `Driver` configuration option does not appear in the Windows user interface for the Simba Trino ODBC Connector. It is accessible only when you use a connection string or configure a connection in macOS or Linux.

Driver

`refresh_token_interval`

Driver

In Windows, the name of the installed connector for (Simba Trino ODBC Connector).

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

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

refresh_token_interval

This option specifies the time interval between the two requests to fetch the access token from the AzureAD. If the value is greater than access token expiry limit, it ignores refresh_token_interval value and continues with the default behavior.

The value ranges from 0 to 65535.

Key Name	Default Value	Required
refresh_token_interval	Half of access token expiry time	No

Third-Party Trademarks

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

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

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

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

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

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

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

All other trademarks are trademarks of their respective owners.