insightsoftware

Simba Salesforce ODBC Data Connector

Installation and Configuration Guide

Version 3.0

November 2025

Contents

Copyright

Contact Us

www.insightsoftware.com

About This Guide

Purpose

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

Knowledge Prerequisites

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

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

The Simba Salesforce ODBC Connector enables Business Intelligence (BI), analytics, and reporting on data that is stored in Salesforce.com. The connector complies with the ODBC 3.80 data standard and adds important functionality such as Unicode, as well as 32- and 64-bit support for high-performance computing environments on all platforms.

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

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

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

Platform and Data source version support

The Simba Salesforce 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.
- 150 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:

- `SimbaSalesforceODBC32.msi` for 32-bit applications
- `SimbaSalesforceODBC64.msi` for 64-bit applications

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

To install the Simba Salesforce ODBC Connector in Windows:

1. Depending on the bitness of your client application, double-click to run **Simba Salesforce <Version Number> 32-bit.msi** or **Simba Salesforce <Version Number> 64-bit.msi**.
2. Click **Next**.
3. Select the check box to accept the terms of the License Agreement if you agree, and then click **Next**.
4. To change the installation location, click **Change**, then browse to the desired folder, and then click **OK**. To accept the installation location, click **Next**.
5. Click **Install**.
6. When the installation completes, click **Finish**.

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

Creating a Data Source Name in Windows

Typically, after installing the Simba Salesforce 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 Salesforce.

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

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

4. Click **Add**.
5. In the Create New Data Source dialog box, select **Simba Salesforce ODBC Connector**and then click **Finish**. The Simba Salesforce ODBC ConnectorDSN Setup dialog box opens.
6. In the **Data Source Name** field, type a name for your DSN.
7. Optionally, in the **Description** field, type relevant details about the DSN.
8. Select the **Auth Type**.
9. For OAuth 2.0 connections:
   a. In the **Client ID** field, type the OAuth 2.0 client ID.
   b. In the **Client Secret** field, type the OAuth 2.0 client secret.
   c. In the **Refresh Token** field, type the refresh token that you obtain from the OAuth 2.0 web server or user-agent flow for authorizing access to Salesforce.
   d. Optionally, in the **Sandbox URL** field, type the Sandbox URL.

10. For Basic Auth connections:
    a. In the **Username** field, type the user name for your Salesforce account.

> **Note:**
>
> When you connect to your data, you will be prompted to provide your Salesforce account credentials. If your user name is saved in the DSN, then you will not need to type it again.

    b. In the **Password** field, type the password corresponding to the user name you typed above.

> **Note:**
>
> For security reasons, passwords are not saved in the DSN; when you connect to your data, you will be prompted to type your password again. Providing your password in the DSN allows you to test your connection.

    c. Optionally, in the **Sandbox URL** field, type the Sandbox URL.

11. If you are using a proxy server, then select the **Use Proxy Server** check box and then do the following:

a. In the **Proxy Host** field, type the host name or IP address of the proxy server.

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

c. In the **Proxy Username** field, type your user name for accessing the proxy server.

d. In the **Proxy Password** field, type the password corresponding to the user name you specified above.

12. To configure TLS settings, do the following:
    a. To specify the trusted CA certificates that you want to use to verify the server, do one of the following:
        - To verify the server using the certificates from a specific `.pem` file, specify the full path to the file in the **Trusted Certificates** field and leave the **Use System Trust Store** check box clear.
        - Or, to use the certificates from the `.pem` file that is installed with the connector, leave the default value in the **Trusted Certificates** field, and leave the **Use System Trust Store** check box clear.
        - Or, to use the certificates from the Windows trust store, select the **Use System Trust Store** check box and leave the **Trusted Certificates** field cleared.
    b. To specify the minimum version of TLS to use, from the **Minimum TLS** drop-down list, select the minimum version of SSL.

Important:

- Salesforce does not support TLS 1.0 or earlier. If you set this option to 1.0, your connection might fail.
- The Simba Salesforce ODBC Connector requires an SSL/TLS connection.

13. To configure advanced connector options, click **Advanced Options**. For more information, see Configuring Advanced Options in Windows.
14. To configure logging behavior for the connector, click **Logging Options**. For more information, see Configuring Logging Options in Windows.
15. To test the connection, click **Test**. Review the results as needed, and then click **OK**.

Important:
If the connection fails, then you might need to provide a security token. To obtain a security token, follow the instructions in the Salesforce documentation located at https://help.salesforce.com/apex/HTViewHelpDoc?id=user_security_token.htm. Copy and paste the security token into the Security Token field, and then test the connection again. You can save the security token in the DSN by selecting the **Save Security Token** check box.

16. To save your settings and close the Simba Salesforce ODBC Driver DSN Setup dialog box, click **OK**.
17. 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 connect to a Salesforce sandbox, select the **Enable Connection To Sandbox URL** check box and then type the URL of the sandbox in the **Sandbox URL** field.
3. To specify the query language that the connector uses to parse queries, select the appropriate **Parse Method** setting.
4. To retrieve data using the Bulk API, select the **Enable Bulk API Queries** check box and then, in the **Bulk API Query At This Many Records** field, type the number of query results at which the connector starts to use the Bulk API.

Note:
If data retrieval fails, then the connector falls back to using the REST API to retrieve the data.

5. If Bulk API queries are enabled, then you can configure the connector to use primary key chunking when retrieving data by doing the following:

a. Select the **Enable Primary Key Chunking** check box.

b. In the **Primary Key Chunking At This Many Records** field, type the number of query results at which the connector starts to use primary key chunking.

c. In the **Primary Key Chunk Size** field, type the number of query results to be included in each chunk.

6. To specify how the connector initiates polls when checking the status of a batch operation in Salesforce, configure the settings in the Bulk API Options area as follows:

   - To initiate polls at a specific time interval, clear the **Enable Backoff When Checking Status** check box and then, in the **Bulk Polling Interval** field, type the amount of time between polls in milliseconds.

   - Or, to initiate polls based on an exponential backoff policy, select the **Enable Backoff When Checking Status** check box and then do the following:

     - In the **Minimum Backoff Delay** field, type the minimum amount of time between polls in milliseconds.

     - In the **Maximum Backoff Delay** field, type the maximum amount of time between polls in milliseconds.

7. To infer metadata based on a small sampling of the data rather than all of the data, select the **Enable Report Metadata Optimization** check box.

Important:
Enabling this option allows the connector to run faster, but the metadata might be less accurate.

8. To specify whether the connector uses field labels or names from Salesforce as the labels in the returned data, do one of the following:

   - To use the field labels from Salesforce as the labels in the returned data, select the **Use Salesforce Labels For Columns** check box.

   - Or, to use the field names from Salesforce as the labels in the returned data, clear the **Use Salesforce Labels For Columns** check box.

Note:
Some client applications require the names and labels in the returned data to match.

9. To return data as SQL_WVARCHAR data instead of SQL_VARCHAR data, select the **Use SQL_WVARCHAR Instead Of SQL_VARCHAR** check box.

Note:
This option applies only to result set columns that the connector would normally return as SQL_VARCHAR columns. It does not convert all columns into SQL_WVARCHAR.

10. To return data as SQL_NUMERIC data instead of SQL_DOUBLE data, select the **Use SQL_NUMERIC For Result Set Columns Of Type SQL_DOUBLE** check box.

Note:
This option only applies to result set columns that the connector would normally return as SQL_ DOUBLE columns. It does not convert all columns into SQL_NUMERIC.

11. To execute reports using the Analytic API, which is also called the Reports and Dashboards REST API, instead of using the report URL, select the **Use Analytic API For Executing Reports** check box.

Note:

- The Analytic API provides metadata which can make the connector more accurate, but is limited to the first 2000 results.
- If this option is selected, the connector uses the Analytic API to execute reports that have IDs. If a report does not have an ID, the connector falls back to using the report URL to execute the report. For more information, see Use Analytic API For Executing Reports.

12. To modify catalog names by removing all invalid SQL-92 identifier characters and replacing all spaces with underscores, select the **Sanitize Catalog Names** check box.
13. To remove catalog names from the TableName arguments in SQLTables and SQLColumns function calls, select the **Strip Catalog Name From Filter Arguments** check box.

Important:
For SQLTables and SQLColumns calls, the connector does not accept TableName arguments that include catalog names. If the TableName arguments include catalog names and the Strip Catalog Name From Filter Arguments check box is clear, the connector does not return any results.

14. To prevent the connector from joining tables during passdown queries, select the **Disable Join Passdown** check box.
15. To allow the connector to use JunctionIdLists, select the **Enable JunctionIdLists** check box.
16. To retrieve more detailed metadata during SQLTables or SQLColumns calls, and use Salesforce data type names instead of SQL data type names when returning the column types, select the **Enable Extended Metadata** check box. For detailed information about the additional metadata that is returned, see Extended Metadata.

Important:
When this option is enabled, connector performance may decrease significantly due to the amount of additional metadata that is retrieved.

17. To save your settings and close the Advanced Options dialog box, click **OK**.

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 SimbaSalesforce 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 Simba Salesforce 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 Salesforce 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**.

   Note: A warning message appears when users without the administrator permissions click **Logging Options**. After acknowledging the warning, the Logging Options dialog opens with all settings disabled (greyed out) to prevent unauthorized modifications.

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. You can type the path into the field, or click **Browse** and then browse to select the folder.
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 Salesforce ODBC Connector produces the following log files at the location you specify in the Log Path field:

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

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

To disable connector logging in Windows:

1. 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. Choose one:
   - If you are using Windows 7 or earlier, click **Start**, then type **regedit** in the **Search** field, and then click **regedit.exe** in the search results.
   - Or, if you are using Windows 8 or later, on the Start screen, type **regedit**, and then click the **regedit** search result.
2. Navigate to the appropriate registry key for the bitness of your connector and your machine:
   - 32-bit System DSNs: **HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\ODBC\ODBC.INI*[DSN Name]***
   - 64-bit System DSNs: **HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI*[DSN Name]***
   - 32-bit and 64-bit User DSNs: **HKEY_CURRENT_USER\SOFTWARE\ODBC\ODBC.INI*[DSN Name]***
3. For each configuration option that you want to configure for the current connection, create a value by doing the following:
   a. If the key name value does not already exist, create it. Right-click the *[DSN Name]* and then select **New > String Value**, type the key name of the configuration option, and then press **Enter**.
   b. Right-click the key name and then click **Modify**.

      To confirm the key names for each configuration option, see Connector Configuration Options.

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

4. Close the Registry Editor.
5. Restart your ODBC application to make sure that the new settings take effect.

Verifying the Connector Version Number in Windows

If you need to verify the version of the Simba Salesforce 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 Salesforce.

2. Click the **Drivers** tab and then find the Simba Salesforce 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, 3.52.10, 3.52.11, or 3.52.12
  - unixODBC 2.3.1

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

To install the Simba Salesforce ODBC Connector in macOS:

1. Double-click **SimbaSalesforceODBC.dmg** to mount the disk image.
2. Double-click **SimbaSalesforce 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/`salesforce 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/salesforceodbc/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 Salesforce 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.salesforceodbc
```

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

Linux Connector

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

The Linux connector is available as an RPM file and as a tarball package.

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:

- 150MB of available disk space
- One of the following ODBC driver managers installed:
  - iODBC 3.52.9, 3.52.10, 3.52.11, or 3.52.12
  - unixODBC 2.3.1

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*.

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

- `SimbaSalesforceODBC-32bit-[Version]-[Release].rpm` for the 32-bit connector
- `SimbaSalesforceODBC-[Version]-[Release].rpm` for the 64-bit connector

The placeholders in the file names are defined as follows:

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

To install the Simba Salesforce 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:

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

```
zypper install [RPMFileName]
```

4. If you received a license file through email, then copy the license file into the `/opt/simba/salesforceodbc/lib/32` or `/opt/simba/salesforceodbc/lib/64` folder, depending on the version of the connector that you installed.

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

Installing the Connector Using the Tarball Package

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

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

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

To install the connector using the tarball package:

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

```
tar  --directory=/opt -zxvf [TarballName]
```

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

   The Simba Salesforce ODBC Connector files are installed in the `/opt/simba/`salesforce directory.

3. If you received a license file through email, then copy the license file into the `opt/simba/salesforceodbc/lib/32` or `opt/simba/salesforceodbc/lib/64` folder, depending on the version of the connector that you installed.

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

Verifying the Connector Version Number in Linux

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

To verify the 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 SimbaSalesforceODBC
  - rpm -qa | grep SimbaSalesforceODBC

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

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 Salesforce 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.salesforceodbc.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 SIMBA_SALESFORCE_ODBC_INI to the full path and file name of the `simba.salesforceodbc.ini` file.

If you are using unixODBC, do the following:

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

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

For unixODBC:

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

export ODBCSYSINI=/usr/local/odbc

export SIMBA_SALESFORCE_ODBC_INI=/etc/simba.salesforceodbc.ini

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

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

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

Configuring ODBC Connections in Non-Windows Machine

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

- Creating a Data Source Name on a Non-Windows Machine
- Configuring a DSN-less Connection in a Non-Windows Machine
- Configuring Logging Options in a Non-Windows Machine
- Testing the Connection on a Non-Windows Machine

Creating a Data Source Name on a Non-Windows Machine

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

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

[ODBC Data Sources]

Sample DSN=Simba Salesforce 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/salesforceodbc/lib/libsfodbc_sbu.dylib

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

   Driver=/opt/simba/salesforceodbc/lib/32/libsfodbc_sb32.so

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

> **Important:**
> If the connection fails when you test it, then you might need to provide a security token for authentication. To obtain a security token, follow the instructions in the Salesforce documentation located at https://help.salesforce.com/apex/HTViewHelpDoc?id=user_security_token.htm.
>
> In the `odbc.ini` file, in the section corresponding to your DSN, set the `SecurityToken` property to the security token that you obtained.

c. 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 Salesforce ODBC Connector, see Connector Configuration Properties.

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 Salesforce:

[ODBC Data Sources]

Sample DSN=Simba Salesforce ODBC Connector

[Sample DSN]

Driver=/Library/simba/salesforceodbc/lib/libsfodbc_sbu.dylib

UID=simba

PWD=simba123

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

[ODBC Data Sources]

Sample DSN=Simba Salesforce ODBC Connector 32-bit

[Sample DSN]

Driver=/opt/simba/salesforceodbc/lib/32/libsfodbc_sb32.so

UID=simba

PWD=simba123

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

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

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

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

To define a connector on a non-Windows machine:

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

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

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

   For example:

   [ODBC Drivers]

   Simba Salesforce 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/salesforceodbc/lib/libsfodbc_sbu.dylib

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

      Driver=/opt/simba/salesforceodbc/lib/32/libsfodbc_sb32.so

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

      For example:

      Description=Simba Salesforce 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 Salesforce ODBC Connector=Installed

   [Simba Salesforce ODBC Connector]

   Description=Simba Salesforce ODBC Connector

   Driver=/Library/simba/salesforceodbc/lib/libsfodbc_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 Salesforce ODBC Connector 32-bit=Installed

   Simba Salesforce ODBC Connector 64-bit=Installed

   [Simba Salesforce ODBC Connector 32-bit]

   Description=Simba Salesforce ODBC Connector (32-bit)

   Driver=/opt/simba/salesforceodbc/lib/32/libsfodbc_sb32.so

   [Simba Salesforce ODBC Connector 64-bit]

   Description=Simba Salesforce ODBC Connector (64-bit)

   Driver=/opt/simba/salesforceodbc/lib/64/libsfodbc_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 on a Non-Windows Machine

Some Salesforce servers are configured to require authentication for access. To connect to an Salesforce server, you must configure the Simba Salesforce ODBC Connector to use the authentication mechanism that matches the access requirements of the server and provides the necessary credentials.

You can set the connection properties for authentication 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.

The following authentication methods are available:

Using User Name And Password

This authentication mechanism requires a user name and a password.

To configure User Name And Password authentication:

1. Set the `UID` property to an appropriate user name for accessing the Salesforce server.
2. Set the `PWD` property to the password corresponding to the user name you provided above.
3. Optionally, to use Basic authentication, set the `Auth_Type` property to the `Basic`.
4. Optionally, set the `URL` property to your Sandbox URL

Using OAuth 2.0

This authentication mechanism requires a client ID, client secret, refresh token, and Sandbox URL.

To configure OAuth 2.0 authentication:

1. Set the `Auth_Type` property to `OAuth_2.0`.
2. Set the `OAuthMechanism` property to `2`.
3. Set the `ClientId` property to your client ID.
4. Set the `ClientSecret` property to your client secret.
5. Set the `RefreshToken` property to your refresh token.
6. Optionally, set the `URL` property to your Sandbox URL

Important:
The `Auth_Type` property currently only supports the values `Basic` or `OAuth_2.0`, and the `OAuthMechanism` property currently only supports the value `2`.

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

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

To enable logging on a non-Windows machine:

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

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

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

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

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

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

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

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

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

If you set the `UseLogPrefix` property to `1`, then each file name is prefixed with `[UserName]_[ProcessID]_`, where *[UserName]* is the user name associated with the connection and *[ProcessID]*

is the process ID of the application through which the connection is made. For more information, see UseLogPrefix.

To disable logging on a non-Windows machine:

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

Testing the Connection on a Non-Windows Machine

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

Using the iODBC Driver Manager

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

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

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

To test your connection using the iODBC driver manager:

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

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

If the connection fails, then you might need to provide a security token. For information about how to provide a security token in your DSN, see Creating a Data Source Name on a Non-Windows Machine.

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] [UserID] [Password]`
  - `iusql [DataSourceName] [UserID] [Password]`

`[DataSourceName]` is the DSN that you are using for the connection. `[UserID]` and `[Password]` are your credentials for accessing the Salesforce data source.

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

If the connection fails, then you might need to provide a security token. For information about how to provide a security token in your DSN, see Creating a Data Source Name on a Non-Windows Machine.

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

Using a Connection String

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

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

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.
- *[OAuthMechanism]* is the OAuth 2.0 authentication mechanism that is used for authentication.
- *[PortNumber]* is the number of the TCP port that the proxy server uses to listen for client connections.
- *[Server]* is the IP address or host name of the proxy server to which you are connecting.
- *[Token]* is the security token that you obtain from Salesforce.com for authorizing access to Salesforce.
- *[YourClientID]* is the client ID that is used for OAuth 2.0 authentication.
- *[YourClientSecret]* is the client secret that is used for OAuth 2.0 authentication.
- *[YourPassword]* is the password corresponding to your Salesforce user name.

- *[YourRefreshToken]* is the refresh token that is used for OAuth 2.0 authentication.
- *[YourUserName]* is the user name for your Salesforce account.

Connecting to Salesforce.com Directly

The following is the format of a DSN-less connection string for a standard connection to Salesforce.com:

Driver=Simba Salesforce ODBC Driver;UID=*[YourUserName]*;
PWD=*[YourPassword]*;

For example:

Driver=Simba Salesforce ODBC Driver;UID=simba;
PWD=simba;

Some Salesforce connections require you to provide a security token. The following is an example of a DSN-less connection string for a standard connection with a security token:

Driver=Simba Salesforce ODBC Driver;UID=simba;
PWD=simba;SecurityToken=abc123;

> **Note:**
>
> To obtain a security token, follow the instructions in the Salesforce documentation located at https://help.salesforce.com/apex/HTViewHelpDoc?id=user_security_token.htm

Connecting to Salesforce.com through a Proxy Server

The following is the format of a DSN-less connection string for connecting to Salesforce.com through a proxy server:

Driver=Simba Salesforce ODBC Driver;UID=*[YourUserName]*;
PWD=*[YourPassword]*;ProxyHost=*[Server]*;ProxyPort=*[PortNumber]*;

For example:

Driver=Simba Salesforce ODBC Driver;UID=simba;
PWD=simba;ProxyHost=192.168.222.160;ProxyPort=8000;

Some Salesforce connections require you to provide a security token. The following is the format of a DSN-less connection string for a proxy server connection with a security token:

Driver=Simba Salesforce ODBC Driver;UID=simba;
PWD=simba;SecurityToken=abc123;
ProxyHost=192.168.222.160;ProxyPort=8000;

Connecting to Salesforce.com using Basic authentication

The following is the format of a DSN-less connection string for connecting to Salesforce.com using Basic authentication:

Driver=SimbaSalesforce ODBC Driver;Auth_Type=*[AuthType]*;UID=*[YourUserName]*;PWD=*[YourPassword]*;

For example:

Driver=SimbaSalesforce ODBC Driver;Auth_Type=Basic;UID=simba;PWD=simba;

Connecting to Salesforce.com using OAuth 2.0

The following is the format of a DSN-less connection string for connecting to Salesforce.com using OAuth 2.0 protocol:

Driver=SimbaSalesforce ODBC Driver;Auth_Type=*[AuthType]*;OAuthMechanism=*[OAuthMechanism]*;ClientId=*[YourClientID]*;ClientSecret=*[YourClientSecret]*;RefreshToken=*[YourRefreshToken]*;

For example:

Driver=SimbaSalesforce ODBC Driver;Auth_Type=OAuth_2.0;OAuthMechanism=2;ClientId=45kn9Pcyq9pr4lvu123pfl4r57;ClientSecret=15kLtgactKYlqOqFxzuzvmFkKo0KM;RefreshToken=CH01pcNn/qFcYwUlJpkF_yyufYrqj4O4g7cdXvGgszT6;

Important:
The `Auth_Type` property currently only supports the values `Basic` or `OAuth_2.0`, and the `OAuthMechanism` property currently only supports the value `2`.

Note:
To obtain a security token, follow the instructions in the Salesforce documentation located at https://help.salesforce.com/apex/HTViewHelpDoc?id=user_security_token.htm.

Features

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

- SQL Connector
- Data Types
- Extended Metadata
- Write-back
- Retrieving Reports
- Security and Authentication

SQL Connector

The SQL Connector feature of the connector allows applications to use normal SQL queries against Salesforce.com, translating standard SQL-92 queries into equivalent Salesforce API calls. This translation allows standard queries that BI tools execute to run against your Salesforce data source.

Data Types

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

Some Salesforce data may be returned differently depending on how the connector is configured:

- If the Use SQL_WVARCHAR instead of SQL_VARCHAR option (the `UseWVarChar` property) is enabled, then data types that are normally returned as SQL_VARCHAR are returned as SQL_WVARCHAR instead. For more information, see Use SQL_WVARCHAR instead of SQL_VARCHAR.
- If the Use SQL_NUMERIC For Result Set Columns Of Type SQL_DOUBLE option (the `UseNumeric` property) is enabled, then data types that are normally returned as SQL_DOUBLE are returned as SQL_NUMERIC instead. For more information, see Use SQL_NUMERIC for Result Set Columns of Type SQL_DOUBLE.
- When the Use Analytic API For Executing Reports option (the `UseAnalyticAPI` property) is disabled, timestamps are returned in the time zone of your locale. Otherwise, timestamps are returned in UTC time. For more information, see Use Analytic API For Executing Reports.

The following table lists the supported data type mappings.

Salesforce Type	SQL Type
AnyType	SQL_VARCHAR or SQL_WVARCHAR

Salesforce Type	SQL Type
Base64	SQL_LONGVARBINARY
Boolean	SQL_BIT
Combobox	SQL_VARCHAR or SQL_WVARCHAR
Currency	SQL_NUMERIC or SQL_DOUBLE
DataCategoryGroupReference	SQL_VARCHAR or SQL_WVARCHAR
Date	SQL_TYPE_DATE **Note:** ■ In CSV reports, Date data is returned as a preformatted string. ■ In versions of the ODBC API earlier than 3.0, this data type is called SQL_DATE.
DateTime	SQL_TYPE_TIMESTAMP **Note:** ■ In CSV reports, DateTime data is returned as a preformatted string. ■ In versions of the ODBC API earlier than 3.0, this data type is called SQL_TIMESTAMP.
Double	SQL_NUMERIC or SQL_DOUBLE
Email	SQL_VARCHAR or SQL_WVARCHAR
Id	SQL_VARCHAR or SQL_WVARCHAR
Int	SQL_INTEGER
JunctionIdListNames	SQL_VARCHAR or SQL_WVARCHAR
MultiPicklist	SQL_VARCHAR or SQL_WVARCHAR
Percent	SQL_NUMERIC or SQL_DOUBLE

Salesforce Type	SQL Type
Phone	SQL_VARCHAR or SQL_WVARCHAR
Picklist	SQL_VARCHAR or SQL_WVARCHAR
Reference	SQL_VARCHAR or SQL_WVARCHAR
String	SQL_VARCHAR or SQL_WVARCHAR
TextArea	SQL_VARCHAR or SQL_WVARCHAR
Time	SQL_TYPE_TIME **Note:** ■ In CSV reports, Time data is returned as a preformatted string. ■ In versions of the ODBC API earlier than 3.0, this data type is called SQL_TIME.
URL	SQL_VARCHAR or SQL_WVARCHAR

Note:
The Byte, Calculated, and MasterRecordId data types are not supported.

Extended Metadata

The Simba Salesforce ODBC Connector can retrieve detailed metadata about Salesforce tables and columns. This information is available only if the Enable Extended Metadata check box is selected or the `ExtendedMetadata` property is set to `1`. When that option is enabled, the connector returns additional metadata as JSON objects in the REMARKS column when you call SQLTables or SQLColumns.

Important:

- If any of the JSON objects in the REMARKS column ends with a `"truncated":true` attribute, this means that the connector has truncated the returned metadata because it exceeds the maximum number of characters that the REMARKS column can contain (2,027,000 characters).
- When this feature is enabled, connector performance may decrease significantly due to the amount of additional metadata that is retrieved.

The following table lists the additional metadata that can be retrieved, the function call that returns the metadata, and the name of the JSON object in which the metadata is returned:

Metadata	Function Call	JSON Object
Table display names	SQLTables	`label`
Column display names	SQLColumns	`label`
Privileges associated with the tables or columns	SQLTables or SQLColumns	`privileges`
Enum values for Picklist columns	SQLColumns	`picklistValues`
Compound field name for Compound fields	SQLColumns	`compoundFieldName`

The privileges associated with a table or column are indicated by the following fields in the `privileges` JSON object:

Field Name	Description
`select`	This field indicates whether the data in the table or column is readable.
`insert`	This field indicates whether the data in the table or column is writeable.
`update`	This field indicates whether the data in the table or column can be updated.
`delete`	This field indicates whether the data in the table or column can be deleted.

For example, when you call SQLColumns on a Picklist column, and the extended metadata feature is enabled, the connector returns a REMARKS column with contents such as the following:

```
{
    "label": "Account Type",
    "privileges" :
    {
        "select": "true",
        "insert": "true",
```

```
            "update": "true",
        },
        "picklistValues":
        [
            {"value": "Analyst",
            "label":" Analyst "},
            {"value": "Competitor",
            "label":" Competitor "} ,
            {"value": "Customer",
            "label":" Customer "} ,
            {"value": "Integrator",
            "label":" Integrator "} ,
            {"value": "Investor",
            "label":" Investor "} ,
            {"value": "Partner",
            "label":" Partner "} ,
            {"value": "Press",
            "label":" Press "} ,
            {"value": "Prospect",
            "label":" Prospect "} ,
            {"value": "Reseller",
            "label":" Reseller "} ,
            {"value": "Other",
            "label":" Other "}
        ]
}
```


This returned metadata indicates the following:

- The column has "Account Type" as its display name.
- You can read, write, and update the data that the column contains. However, you cannot delete the data.
- The column contains picklist values such as Analyst, Competitor, Customer, and so on.

Write-back

The Simba Salesforce ODBC Connector supports Data Manipulation Language (DML) statements such as INSERT, UPDATE, and DELETE. Write-back operations use the Salesforce Bulk API for loading and deleting data.

Binary data types are not supported for INSERT, UPDATE, and DELETE statements because the Salesforce Bulk API currently does not support binary data types.

Note:
Data Definition Language (DDL) statements such as CREATE, ALTER, and DROP are not supported by the connector.

Retrieving Reports

The Simba Salesforce ODBC Connector supports Salesforce reports by treating them as stored procedures. When you call the SQLProcedures ODBC function without setting any parameters, the connector returns a list of reports from Salesforce.com.

You can retrieve a report by calling the report name using the same syntax as you would for stored procedures. For example, to retrieve a Salesforce report named "Activity Report", you would execute the following statement:

{call "Activity Report"}

You can then work with the data in the report as you would with any other Salesforce.com data.

Important:
Reports that are designed to display in HTML might not display properly when retrieved through the connector.

Security and Authentication

To protect data from unauthorized access, Salesforce requires that all connections be authenticated with either the OAuth 2.0 protocol or with user credentials, which must be encrypted using TLS 1.1 or later.

The connector provides mechanisms that allow you to complete an OAuth 2.0 authentication flow using a client ID, client secret, and refresh token. The connector retrieves a token based on the account credentials specified in your DSN or connection string, and then uses the token to authenticate the connection to Salesforce. For detailed configuration instructions, see Using a Connection String or Creating a Data Source Name on a Non-Windows Machine

Additionally, the Simba Salesforce ODBC Connector automatically applies one-way TLS authentication to all connections, and provides a mechanism that enables you to authenticate your connection using your Salesforce user credentials. For detailed configuration instructions, see Creating a Data Source Name in Windows or Creating a Data Source Name on a Non-Windows Machine.

Connector Configuration Properties

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

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

- Simba Salesforce Configuration
- Advanced Options
- Logging Options

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

Configuration Options Appearing in the User Interface

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

- Auth Type
- Bulk API Query At This Many Records
- Bulk Polling Interval
- Client ID
- Client Secret
- Disable Join Passdown
- Enable Backoff When Checking Status
- Enable Bulk API Queries
- Enable Connection to Sandbox URL
- Enable JunctionIDLists
- Enable Primary Key Chunking
- Enable Report Metadata Optimization
- Password
- Primary Key Chunk Size
- Primary Key Chunking At This Many Records
- Proxy Host
- Proxy Password
- Proxy Port
- Proxy Username
- Refresh Token
- Sandbox URL
- Sanitize Catalog Names
- Security Token

Auth Type

This option specifies the authentication method to use.

Set this property to `Basic` or `OAuth_2.0`.

Key Name	Default Value	Required
`Auth_Type`	`Basic`	Yes, if authenticating through Basic authentication or OAuth 2.0.

Bulk API Query At This Many Records

If the number of results returned by a query is greater than or equal to this value, and the Enable Bulk API Queries option (the `EnableBulkQuery` property) is enabled, then the connector attempts to use the Bulk API for data retrieval. If data retrieval fails, then the connector falls back to using the REST API to retrieve the data.

Key Name	Default Value	Required
`BulkQueryThreshold`	`30000`	No

Bulk Polling Interval

The time in milliseconds that the connector waits between polls when checking the status of a batch operation in Salesforce. The value must be an integer between 1 and 600,000 (inclusive).

For information about configuring the connector to use an exponential backoff policy instead of initiating polls at a specific time interval, see Enable Backoff When Checking Status. If the Enable Backoff When Checking Status (`EnableBulkBackoff`) option is enabled, it takes precedence over the Bulk Polling Interval (`BulkPollInterval`) setting.

Key Name	Default Value	Required
`BulkPollInterval`	`1000`	No

Client ID

The OAuth 2.0 client ID, which is used to generate the refresh token.

Key Name	Default Value	Required
`ClientId`	None	Yes, if authenticating through OAuth 2.0.

Client Secret

The OAuth 2.0 client secret, which is used to generate the refresh token.

Key Name	Default Value	Required
`ClientId`	None	Yes, if authenticating through OAuth 2.0.

Disable Join Passdown

This option specifies whether to join tables during a query passdown.

The connector issues SOQL queries to Salesforce for data retrieval. In SOQL, a given table can be joined at most once. However, a user can specify a SQL query that joins a table with itself any number of times. This may result in that table being passed down repeatedly every time it is joined, which can cause performance issues with large tables.

- Enabled (`1`): The connector does not join tables during a query passdown. Instead, the connector retrieves the entire table that is to be joined, and reuses that table during the query.
- Disabled (`0`): The connector generates SOQL queries that capture any necessary joins during a query passdown.. This may result in redundant fetches being executed if the same table is joined repeatedly.

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

Enable Backoff When Checking Status

When this option is enabled (1), the connector uses an exponential backoff policy when polling the status of a batch operation in Salesforce.

You can specify the minimum and maximum time intervals between poll times by setting the following properties:

- Maximum Backoff Delay
- Minimum Backoff Delay

For information about configuring the connector to initiate polls at a specific time interval instead of using an exponential backoff policy, see Bulk Polling Interval. If the Enable Backoff When Checking Status (EnableBulkBackoff) option is enabled, it takes precedence over the Bulk Polling Interval (BulkPollInterval) setting.

Key Name	Default Value	Required
EnableBulkBackoff	Selected (1)	No

Enable Bulk API Queries

When this option is enabled (1), the connector attempts to use the Bulk API to retrieve data. The query must be executable through the Bulk API, and the number of results returned must be equal to or greater than the value set for the Bulk API Query At This Many Records (BulkQueryThreshold) option. If data retrieval fails, then the connector falls back to using the REST API to retrieve the data.

Note:
This option is only available when the connector parses queries as SQL statements. For more information see Parse Method.

Key Name	Default Value	Required
EnableBulkQuery	Selected (1)	No

Enable Connection to Sandbox URL

When this option is enabled (1), the connector connects to the Salesforce sandbox specified in the Sandbox URL field.

When this option is disabled, the connector does not connect to the sandbox.

Key Name	Default Value	Required
N/A	Clear (0)	No

Enable Extended Metadata

This option specifies whether the connector returns additional Salesforce metadata in the REMARKS column when you call SQLTables or SQLColumns, and returns Salesforce data type names instead of

SQL data type names in the TYPE_NAME column when you call SQLColumns.

- Enabled (`1`): The connector returns additional metadata, and uses Salesforce data type names when returning the column types. For detailed information about the additional metadata that is returned, see Extended Metadata.
- Disabled (`0`): The connector does not return additional metadata, and uses SQL data type names when returning the column types.

Important:
When this option is enabled, connector performance may decrease significantly due to the amount of additional metadata that is retrieved.

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

Enable JunctionIDLists

When this option is enabled (`1`), the connector can use Salesforce JunctionIdLists.

Important:
Due to a limitation in Salesforce, in some cases an object that contains a JunctionIdList does not return all of its data. For more information, see the Salesforce documentation.

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

Enable Primary Key Chunking

When this option is enabled (`1`), the connector attempts to use the Bulk API with primary key chunking enabled when retrieving data. If data retrieval fails, then the connector falls back to using the REST API to retrieve the data.

The connector uses primary key chunking only when all of the following requirements are met:

- The query can be executed with primary key chunking.
- The Enable Bulk API Queries (`EnableBulkQuery`) option must be enabled. See Enable Bulk API Queries for more details.
- The Enable Primary Key Chunking (`EnablePKChunking`) option is enabled.
- The number of results is greater than or equal to the value specified for the Bulk API Query At This Many Records (`BulkQueryThreshold`) option. See Bulk API Query At This Many Records for more details.

- The number of results is greater than or equal to the value specified for the Primary Key Chunking At This Many Records (`PKChunkThreshold`) option. See Primary Key Chunking At This Many Records for more details.

The chunk size used is dependent on the Primary Key Chunk Size (`PKChunkSize`) setting. See Primary Key Chunk Size for more details.

Note:
This option is only available when the connector parses queries as SQL statements. For more information see Parse Method.

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

Enable Report Metadata Optimization

When this option is enabled (the key is set to `1` or `LIGHT`), the connector infers metadata based on a small sampling of data rather than all of the data.

Important:
Enabling this option allows the connector to run faster, but the metadata might be less accurate.

When this option is disabled (the key is set to `0` or `FULL`), the connector infers metadata based on all of the data.

Key Name	Default Value	Required
`MetadataLevel`	Clear (`0` or `FULL`)	No

Locale

The locale to use for error messages.

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

Log Level

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

Important:

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

Set the property to one of the following values:

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

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

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

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

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

Log Path

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

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

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

Max File Size

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

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

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

Key Name	Default Value	Required
LogFileSize	20971520	No

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

Maximum Backoff Delay

The maximum amount of time in milliseconds that the connector waits between polls, when the connector is configured to use an exponential backoff policy to poll the status of a batch operation in Salesforce. This option is applicable only when the Enable Backoff When Checking Status (EnableBulkBackoff) option is enabled.

The value must be an integer between 2 and 600,001 (inclusive), and it must be higher than the value set for the Minimum Backoff Delay (BulkBackoffMinDelay) option.

Key Name	Default Value	Required
BulkBackoffMaxDelay	60000	No

Minimum Backoff Delay

The minimum amount of time in milliseconds that the connector waits between polls, when the connector is configured to use an exponential backoff policy to poll the status of a batch operation in

Salesforce. This option is applicable only when the Enable Backoff When Checking Status (`EnableBulkBackoff`) option is enabled.

The value must be an integer between 1 and 600,000 (inclusive), and it must be lower than the value set for the Maximum Backoff Delay (`BulkBackoffMaxDelay`) option.

Key Name	Default Value	Required
`BulkBackoffMinDelay`	`1000`	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.

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

Important: Salesforce does not support TLS 1.0 or earlier. If you set this option to 1.0, your connection might fail.

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

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

Parse Method

The query language that the connector uses to parse queries.

Select one of the following settings, or set the key to one of the values in the parentheses:

- Attempt To Parse Queries As SOQL Only (`0` or `SOQL_ONLY`)
- Attempt To Parse Queries As SQL Only (`1` or `SQL_ONLY`)
- Attempt To Parse Queries As SOQL First, Then SQL (`2` or `SOQL_FIRST`)
- Attempt To Parse Queries In SQL First, Then SOQL (`3` or `SQL_FIRST`)

Be aware that the SOQL_FIRST and SQL_FIRST parse methods may lead to different behavior for similar queries. This can occur when the connector switches between the two modes when trying to find which query language can support the inputted query. The behavior will be consistent within the same query language, however even small changes to a query can cause the connector to change which language it uses to execute the query, causing different results to be returned.

For example, SOQL and SQL handle comparisons against null values differently. SQL will return an unknown state if a comparison operator (such as = or >) is used with a null value and the result will contain zero rows. SOQL will allow such a comparison and will return results. For example, you issue

the query `SELECT Name FROM Account WHERE NumberOfEmployees = NULL`. This query is valid SOQL, so if you are using the SOQL_FIRST mode the connector executes it in SOQL and returns values containing all non-null values. You then 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. The connector executes it in SQL and returns zero values as specified by the SQL specification. Both queries are similar, but the small difference in the query structure results in a different language being used to execute the queries. This changes the results.

If you are expecting a certain behavior, then use either the SOQL_ONLY mode or the SQL_ONLY mode.

Key Name	Default Value	Required
`ParseMethod`	Attempt To Parse Queries As SOQL First, Then SQL (`2` or `SOQL_FIRST`)	No

Password

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

Note: For security reasons, passwords are not saved in the DSN. When you connect to your data, you are prompted to type your password again. Providing your password in the DSN allows you to test your connection in the SimbaSalesforce ODBC Connector DSN Setup dialog box.

Key Name	Default Value	Required
`PWD`	None	Yes

Primary Key Chunk Size

The number of results to include in each chunk, when query results are retrieved using primary key chunking. The maximum value is 250000.

Key Name	Default Value	Required
`PKChunkSize`	`100000`	No

Primary Key Chunking At This Many Records

The number of query results at which the connector starts to use the Bulk API with primary key chunking for data retrieval.

The connector uses primary key chunking only when all of the following requirements are met:

- The query can be executed with primary key chunking.
- The Enable Bulk API Queries (`EnableBulkQuery`) option must be enabled. See Enable Bulk API Queries for more details.
- The Enable Primary Key Chunking (`EnablePKChunking`) option is enabled. See Enable Primary Key Chunking for more details.
- The number of results is greater than or equal to the value specified for the Bulk API Query At This Many Records (`BulkQueryThreshold`) option. See Bulk API Query At This Many Records for more details.
- The number of results is greater than or equal to the value specified for the Primary Key Chunking At This Many Records (`PKChunkThreshold`) option.

If data retrieval fails, then the connector falls back to using the REST API to retrieve the data.

Key Name	Default Value	Required
`PKChunkThreshold`	`100000`	No

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 proxy server.

Proxy Password

The password that you use to access the proxy server.

Key Name	Default Value	Required
`ProxyPassword`	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 Username

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

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

Refresh Token

The refresh token that you obtain from the OAuth 2.0 web server or user-agent flow for authorizing access to Salesforce.

Key Name	Default Value	Required
`RefreshToken`	None	Yes, if authenticating through OAuth 2.0.

Sandbox URL

The URL for connecting to a Salesforce sandbox.

Key Name	Default Value	Required
`URL`	`https://login.salesforce.com`	No

Sanitize Catalog Names

When this option is enabled (`1`), the connector modifies catalog names by removing all invalid SQL-92 identifier characters and replacing all spaces with underscores.

When this option is disabled (`0`), the connector does not modify catalog names.

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

Security Token

Your security token for accessing Salesforce.com. To obtain a security token, follow the instructions in the Salesforce documentation located at https://help.salesforce.com/apex/HTViewHelpDoc?id=user_security_token.htm.

Important:
Some connections require a security token, while others do not. Only provide a security token if your connection fails without it.

Key Name	Default Value	Required
SecurityToken	None	No

Strip Catalog Name from Filter Arguments

This option specifies whether the connector removes catalog names from the TableName arguments in SQLTables and SQLColumns calls.

Important:
For SQLTables and SQLColumns calls, the connector does not accept TableName arguments that include catalog names. If the TableName arguments include catalog names and this option is disabled, the connector does not return any results.

- Enabled (1): The connector removes catalog names from the TableName arguments in SQLTables and SQLColumns calls.
- Disabled (0): The connector does not remove any catalog names.

Key Name	Default Value	Required
StripCatalogName	Clear (0)	No

Trusted Certificates

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

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

Important:
The Simba Salesforce ODBC Connector requires an SSL/TLS connection.

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

Use Analytic API For Executing Reports

This option determines whether the connector executes reports using the Analytic API, which is also called the Reports and Dashboards REST API.

The Analytic API provides metadata about the data types for columns and fields, which makes the connector more accurate. However, the Analytic API only returns the first 2000 results. If you are using the Analytic API, you might need to filter reports to narrow the results.

Note:
The Analytic API only executes reports that have IDs. If a report does not have an ID, the connector falls back to using the report URL to execute the report.

If the Analytic API is not used, the connector is not limited to 2000 results. However, the connector samples the reports to estimate the data types for columns and fields, which might lead to less accurate results.

- Enabled (`1`): The connector executes reports using the Analytic API.
- Disabled (`0`): The connector executes reports using the report URL.

Key Name	Default Value	Required
`UseAnalyticAPI`	Selected (`1`)	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 to the main Salesforce.com server.

Key Name	Default Value	Required
N/A	Clear (`0`)	No

Use Salesforce Labels for Columns

When this option is enabled (`1`), the connector uses the field names and labels from Salesforce as the names and labels in the returned data, respectively.

When this option is disabled (`0`), the connector uses the field names from Salesforce as both the names and the labels in the returned data.

Note:
Some client applications require the name and label in the returned data to match.

Key Name	Default Value	Required
UseLabel	Clear (0)	No

Use SQL_NUMERIC for Result Set Columns of Type SQL_DOUBLE

When this option is enabled (1), the connector returns data as SQL_NUMERIC data instead of SQL_DOUBLE data.

When this option is disabled (0), the connector returns data as SQL_DOUBLE data.

Note:
This option applies only to result set columns that the connector would normally return as SQL_DOUBLE columns. It does not convert all columns into SQL_NUMERIC.

Key Name	Default Value	Required
UseNumeric	Clear (0)	No

Use SQL_WVARCHAR instead of SQL_VARCHAR

This option specifies how data types are mapped to SQL.

- Enabled (1): The connector returns data as SQL_WVARCHAR data instead of SQL_VARCHAR data.
- Disabled (0): The connector returns data as SQL_VARCHAR data.

Note:
This option applies only to result set columns that the connector would normally return as SQL_VARCHAR columns. It does not convert all columns into SQL_WVARCHAR.

Key Name	Default Value	Required
UseWVarChar	Clear (0)	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

User

The user name for your Salesforce account.

Key Name	Default Value	Required
UID	None	Yes

Configuration Options Having Only Key Names

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

- ApiVersion
- AutoLogout
- BulkBatchSize
- Driver
- EnableTransactions
- InitialJunctionIdListFetchSize
- Locale
- MaxJunctionIdListFetchRetries
- OAuthMechanism
- PKChunkMaxMem
- QueryAll
- SimulateTransactions
- UIDDomain
- UseDefaultSystemProxy
- UseLogPrefix

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

ApiVersion

The Salesforce API version that you want to use for the connection. You must specify the full version number in decimal format. For example, API version 45 must be specified as `45.0`.

Key Name	Default Value	Required
`ApiVersion`	`45.0`	No

AutoLogout

When this option is enabled (`1`), the Salesforce connection is logged out when the connector closes the connection.

When this option is disabled (`0`), the Salesforce connection is not logged out when the connector closes the connection.

Important:
To enable support for multithreading, you must disable this option.

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

BulkBatchSize

The maximum number of rows contained in a single Bulk API call when executing DML. The maximum value is 10000.

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

Client ID

The OAuth 2.0 client ID, which is used to generate the refresh token.

Key Name	Default Value	Required
`ClientId`	None	Yes, if authenticating through OAuth 2.0.

Client Secret

The OAuth 2.0 client secret, which is used to generate the refresh token.

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

Key Name	Default Value	Required
		OAuth 2.0.

Driver

In Windows, the name of the installed connector for (`Simba Salesforce 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 Salesforce ODBC Connector` when installed in Windows, or the absolute path of the connector shared object file when installed on a non-Windows machine.	Yes

EnableTransactions

When this option is enabled (`1`), the connector simulates transactions for all connections, ensuring compatibility with applications that require transaction support. Simulated transactions are not executed.

This property applies to all connections that use the connector, so it is configured differently compared to properties that are set on a per-connection basis. You cannot set `EnableTransactions` in connection strings.

in Windows machines, you must set this property in one of the following registry keys:

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

On non-Windows machines, you must set this property in the `simba.salesforceodbc.ini` file instead of the `odbc.ini` file.

Note:
For information about configuring the connector to only simulate transactions for specific connections, see SimulateTransactions. If the `SimulateTransactions` key is set, it takes precedence over the `EnableTransactions` setting.

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

InitialJunctionIdListFetchSize

The initial maximum number of JunctionIdList rows that are fetched at one time. The minimum value is 1.

Important:
In some cases, the number of rows containing JunctionIdList data being fetched is too large. When this happens, the connector reduces the fetch size by half and retries the fetch. This continues until the number of rows is no longer too large, or until the maximum number of retry attempts is reached. For more information about retry attempts, see MaxJunctionIdListFetchRetries

Key Name	Default Value	Required
`InitialJunctionIdListFetchSize`	`450`	No

MaxJunctionIdListFetchRetries

The maximum number of retry attempts with reduced fetch size that are completed if InitialJunctionIdListFetchSize is too large.

Key Name	Default Value	Required
`MaxJunctionIdListFetchRetries`	`6`	No

OAuthMechanism

The OAuth 2.0 authentication mechanism used to authenticate the connector.

Set this property to `2`.

Key Name	Default Value	Required
`OAuthMechanism`	`2`	Yes, if authenticating through OAuth 2.0.

PKChunkMaxMem

The maximum size, in bytes, of a Primary Key Chunk batch.

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

QueryAll

When this option is enabled (`1`), Salesforce users will be able to conduct searches for deleted records by including the parameter `isDeleted=true`.

When this option is disabled (0), users will not be able to search for deleted records.

Key Name	Default Value	Required
QueryAll	0	No

SimulateTransactions

When this option is enabled (1), the connector simulates transactions for the connection, ensuring compatibility with applications that require transaction support. Simulated transactions are not executed.

This setting only applies to the current connection. For information about configuring the connector to simulate transactions for every connection, see EnableTransactions. If the SimulateTransactions property is set, it takes precedence over the EnableTransactions setting.

Key Name	Default Value	Required
SimulateTransactions	0	No

UIDDomain

The domain in your Salesforce user name.

For example, if your Salesforce user name is myuser@myorganization.com, then you would set the UID key to myuser and set the UIDDomain key to myorganization.com.

Use this configuration option if your ODBC application does not allow the at sign (@) to be part of the user name.

Key Name	Default Value	Required
UIDDomain	None	No

UseDefaultSystemProxy

This option specifies whether the connector uses the proxy settings from the HTTP_PROXY environment variable.

- Enabled (1): The connector first attempts to read the proxy settings from the default system settings, and if it does not find a proxy there, it reads the proxy settings from the connection string.
- Disabled (0): The connector reads the proxy settings only from the connection string.

For information about enabling the Default System Proxy, see **Connecting to Salesforce.com through a Proxy Server** section under Using a Connection String.

Key Name	Default Value	Required
UseDefaultSystemProxy	False	No

UseLogPrefix

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

Set the property to one of the following values:

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

  For example, if you are connecting as a user named "jdoe" and using the connector in an application with process ID 7836, the generated log files would be named `jdoe_7836_simbasalesforceodbcdriver.log` and `jdoe_7836_simbasalesforceodbcdriver_connection_[Number].log`, where *[Number]* is a number that identifies each connection-specific log file.

- `0`: The connector does not include the prefix in log file names.

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

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

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

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

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

Third-Party Trademarks

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

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

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

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

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

Salesforce and Salesforce.com are trademarks or registered trademarks of Salesforce.com, Inc. or its subsidiaries in Canada, the United States and/or other countries.

All other trademarks are trademarks of their respective owners.