



Simba Apache Spark ODBC Data Connector

Installation and Configuration Guide

Version 2.9.4

October 2025

Copyright

This document was released in October 2025.

Copyright ©2014-2025 insightsoftware. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission from insightsoftware.

The information in this document is subject to change without notice. insightsoftware strives to keep this information accurate but does not warrant that this document is error-free.

Any insightsoftware product described herein is licensed exclusively subject to the conditions set forth in your insightsoftware license agreement.

Simba, the Simba logo, SimbaEngine, and Simba Technologies are registered trademarks of Simba Technologies Inc. in Canada, the United States and/or other countries. All other trademarks and/or servicemarks are the property of their respective owners.

All other company and product names mentioned herein are used for identification purposes only and may be trademarks or registered trademarks of their respective owners.

Information about the third-party products is contained in a third-party-licenses.txt file that is packaged with the software.

Contact Us

www.insightsoftware.com

Contents

| | |
|--|----|
| Copyright | 2 |
| Contents | 3 |
| About This Guide | 7 |
| Purpose | 7 |
| Audience | 7 |
| Knowledge Prerequisites | 7 |
| Document Conventions | 7 |
| About the Simba Apache Spark ODBC Connector | 8 |
| Windows Connector | 9 |
| Windows System Requirements | 9 |
| Installing the Connector in Windows | 9 |
| Creating a Data Source Name in Windows | 10 |
| Creating a File DSN in Windows | 12 |
| Configuring a DSN-less Connection in Windows | 13 |
| Configuring AOSS Options in Windows | 16 |
| Configuring Authentication in Windows | 17 |
| Configuring Advanced Options in Windows | 27 |
| Configuring a Proxy Connection in Windows | 29 |
| Configuring HTTP Options in Windows | 30 |
| Configuring SSL Verification in Windows | 30 |
| Configuring Server-Side Properties in Windows | 32 |
| Configuring Logging Options in Windows | 33 |

| | |
|--|-----------|
| Configuring Kerberos Authentication for Windows | 35 |
| Verifying the Connector Version Number in Windows | 39 |
| macOS Connector | 40 |
| macOS System Requirements | 40 |
| Installing the Connector in macOS | 40 |
| Verifying the Connector Version Number in macOS | 41 |
| Linux Connector | 42 |
| Linux System Requirements | 42 |
| Installing the Connector Using the RPM File | 43 |
| Installing the Connector Using the Tarball Package | 44 |
| Installing the Connector on Debian | 44 |
| Verifying the Connector Version Number in Linux | 45 |
| AIX Connector | 47 |
| AIX System Requirements | 47 |
| Installing the Connector on AIX | 47 |
| Verifying the Connector Version Number on AIX | 48 |
| Solaris Connector | 49 |
| Solaris System Requirements | 49 |
| Installing the Connector on Solaris | 49 |
| Verifying the Connector Version Number on Solaris | 50 |
| Configuring the ODBC Driver Manager in Non-Windows Machines | 51 |
| Specifying ODBC Driver Managers in Non-Windows Machines | 51 |
| Specifying the Locations of the Connector Configuration Files | 52 |

| | |
|---|-----------|
| Configuring ODBC Connections in Non-Windows Machine | 54 |
| Creating a Data Source Name on a Non-Windows Machine | 54 |
| Configuring a DSN-less Connection in a Non-Windows Machine | 57 |
| Configuring DataStax AOSS Service Discovery on a Non-Windows Machine | 59 |
| Configuring Authentication on a Non-Windows Machine | 60 |
| Configuring SSL Verification in a Non-Windows Machine | 67 |
| Configuring Server-Side Properties on a Non-Windows Machine | 68 |
| Configuring Logging Options in a Non-Windows Machine | 68 |
| Setting Connector-Wide Configuration Options on a Non-Windows Machine | 70 |
| Testing the Connection in Non-Windows Machine | 70 |
| Authentication Mechanisms | 72 |
| Shark Server | 72 |
| Spark Thrift Server on an HDInsight Distribution | 72 |
| Spark Thrift Server on a non-HDInsight Distribution | 72 |
| Using a Connection String | 74 |
| DSN Connection String Example | 74 |
| DSN-less Connection String Examples | 74 |
| Features | 78 |
| SQL Connector for HiveQL | 78 |
| Data Types | 79 |
| Timestamp Function Support | 79 |
| Catalog and Schema Support | 80 |
| spark_system Table | 80 |

| | |
|---|------------|
| Server-Side Properties | 80 |
| Get Tables With Query | 81 |
| Active Directory | 81 |
| Write-back | 81 |
| Dynamic Service Discovery using DataStax AOSS | 81 |
| Security and Authentication | 82 |
| Connector Configuration Options | 83 |
| Configuration Options Appearing in the User Interface | 83 |
| Configuration Options Having Only Key Names | 115 |
| Third-Party Trademarks | 128 |

About This Guide

Purpose

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

Knowledge Prerequisites

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

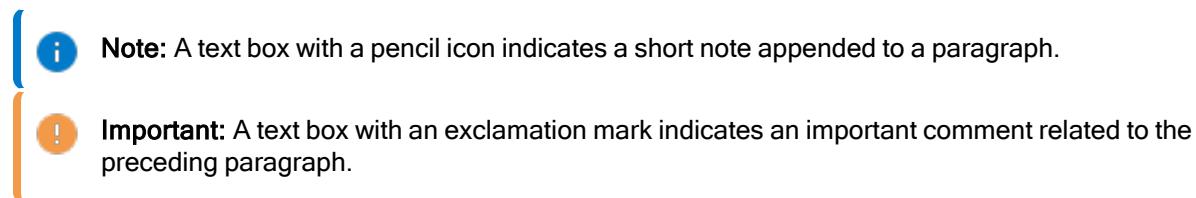
- Familiarity with the platform on which you are using the Simba Apache Spark ODBC Connector
- Ability to use the data source to which the Simba Apache Spark ODBC Connector is connecting
- An understanding of the role of ODBC technologies and driver managers in connecting to a data source
- Experience 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 Apache Spark ODBC Connector

The Simba Apache Spark ODBC Connector is used for direct SQL and HiveQL access to Apache Hadoop / Spark distributions, enabling Business Intelligence (BI), analytics, and reporting on Hadoop-based data. The connector efficiently transforms an application's SQL query into the equivalent form in HiveQL, which is a subset of SQL-92. If an application is Spark-aware, then the connector is configurable to pass the query through to the database for processing. The connector interrogates Spark to obtain schema information to present to a SQL-based application. Queries, including joins, are translated from SQL to HiveQL. For more information about the differences between HiveQL and SQL, see [SQL Connector for HiveQL](#).

The Simba Apache Spark ODBC Connector complies with the ODBC 3.80 data standard and adds important functionality such as Unicode and 32- and 64-bit support for high-performance computing environments.

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 Apache Spark ODBC Connector is available for Microsoft® Windows®, Linux, and macOS platforms.



Note: The AIX and Solaris connectors are not available through the Simba website. To get these connectors, contact the Sales & Solutions team:

- Phone number: +1.604.633.0008 ext 2
- Email: solutions@simba.com



Note: This is the most up-to-date version of the *Installation and Configuration Guide*, for use with version 2.9.4 of the connector. If you are using an older version of the connector, certain features may not be available and certain settings may behave in unexpected ways. Please consult the PDF version of the *Installation and Configuration Guide* that was installed with your connector.

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



Note: For basic configuration instructions that allow you to quickly set up the Windows connector so that you can evaluate and use it, see the *Simba ODBC Connectors Quick Start Guide for Windows*. The Quick Start Guide also explains how to use the connector in various applications.

Windows Connector

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

Windows System Requirements

The Simba Apache Spark ODBC Connector supports Apache Spark versions 3.1.2 through 3.5.0.

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

- Administrator rights on your machine.
- A machine that meets the following system requirements:
 - One of the following operating systems:
 - Windows 11 or 10
 - Windows Server 2025, 2022, 2019, 2016, or 2012
 - 100 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

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:

- Simba Spark 2.9 32-bit.msi for 32-bit applications
- Simba Spark 2.9 64-bit.msi for 64-bit applications

To install the Simba Apache Spark ODBC Connector in Windows:

1. Depending on the bitness of your client application, double-click to run **Simba Spark 2.9 32-bit.msi** or **Simba Spark 2.9 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 Apache Spark ODBC Connector, you need to create a Data Source Name (DSN). A DSN is a data structure that stores connection information so that it can be used by the connector to connect to Spark.

Alternatively, you can specify connection settings in a connection string or as connector-wide settings. Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

The following instructions describe how to create a DSN. For information about specifying settings in a connection string, see [Using a Connection String](#). For information about connector-wide settings, see [Configuring a DSN-less Connection in Windows](#).

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 Spark.

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

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



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

4. Click **Add**.
5. In the Create New Data Source dialog box, select **Simba Apache Spark ODBC Connector** and then click **Finish**. The Simba Apache Spark ODBC Connector Spark DSN Setup dialog box opens.
6. In the **Data Source Name** field, type a name for your DSN.
7. Optionally, in the **Description** field, type relevant details about the DSN.

8. From the **Spark Server Type** drop-down list, select the appropriate server type for the version of Spark that you are running:
 - If you are running Shark 0.8.1 or earlier, then select **SharkServer**.
 - If you are running Shark 0.9, or Spark 1.1 or later, then select **SparkThriftServer**.
9. Specify whether the connector uses the DataStax AOSS service when connecting to Spark, and provide the necessary connection information:
 - To connect to Spark without using the DataStax AOSS service, do the following:
 - From the **Service Discovery Mode** drop-down list, select **No Service Discovery**.
 - In the **Host(s)** field, type the IP address or host name of the Spark server.
 - In the **Port** field, type the number of the TCP port that the Spark server uses to listen for client connections.
 - Or, to discover Spark services via the DataStax AOSS service, do the following:
 - From the **Service Discovery Mode** drop-down list, select **AOSS**.
 - In the **Host(s)** field, type a comma-separated list of AOSS endpoints. Use the following format, where *[AOSS_Endpoint]* is the IP address or host name of the AOSS endpoint, and *[AOSS_Port]* is the number of the TCP port that the AOSS endpoint uses to listen for client connections:
[AOSS_Endpoint1]:[AOSS_Port1],[AOSS_Endpoint2]:[AOSS_Port2]
 - If the AOSS endpoints require different authentication and SSL settings than the Spark service that you are connecting to, click **Service Discovery Options** and configure the options in the AOSS Options dialog box as needed. For more information, see [Configuring AOSS Options in Windows](#).

10. In the **Database** field, type the name of the database schema to use when a schema is not explicitly specified in a query.



Note: You can still issue queries on other schemas by explicitly specifying the schema in the query. To inspect your databases and determine the appropriate schema to use, type the `show databases` command at the Spark command prompt.

11. In the Authentication area, configure authentication as needed. For more information, see [Configuring Authentication in Windows](#).



Note: Shark Server does not support authentication. Most default configurations of Spark Thrift Server require User Name authentication. To verify the authentication mechanism that you need to use for your connection, check the configuration of your Hadoop / Spark distribution. For more information, see [Authentication Mechanisms](#).

12. Optionally, if the operations against Spark are to be done on behalf of a user that is different than the authenticated user for the connection, type the name of the user to be delegated in the **Delegation UID** field.



Note: This option is applicable only when connecting to a Spark Thrift Server instance that supports this feature.

13. From the **Thrift Transport** drop-down list, select the transport protocol to use in the Thrift layer.



Note: For information about how to determine which Thrift transport protocols your Spark server supports, see [Authentication Mechanisms](#).

14. If the Thrift Transport option is set to HTTP, then to configure HTTP options such as custom headers, click **HTTP Options**. For more information, see [Configuring HTTP Options in Windows](#).

15. To configure the connector to connect to Spark through a proxy server, click **Proxy Options**. For more information, see [Configuring a Proxy Connection in Windows](#).

16. To configure client-server verification over SSL, click **SSL Options**. For more information, see [Configuring SSL Verification in Windows](#).



Note: If you selected User Name or Windows Azure HDInsight Emulator as the authentication mechanism, SSL is not available.

17. To configure advanced connector options, click **Advanced Options**. For more information, see [Configuring Advanced Options in Windows](#).

18. To configure server-side properties, click **Advanced Options** and then click **Server Side Properties**. For more information, see [Configuring Server-Side Properties in Windows](#).

19. To configure logging behavior for the connector, click **Logging Options**. For more information, see [Configuring Logging Options in Windows](#).

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



Note: If the connection fails, then confirm that the settings in the Simba Spark ODBC Driver DSN Setup dialog box are correct. Contact your Spark server administrator as needed.

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

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

Creating a File DSN in Windows

Typically, after installing the Simba Apache Spark ODBC Connector, you need to create a Data Source Name (DSN). A DSN is a data structure that stores connection information so that it can be used by the connector to connect to Spark.

To configure a File DSN 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 Spark.

2. In the ODBC Data Source Administrator, click the Drivers tab, and then scroll down as needed to confirm that the Simba Apache Spark ODBC Connector appears in the alphabetical list of ODBC drivers that are installed on your system.
3. Click the **File DSN** tab.
4. Click **Add**.
5. In the Create New Data Source dialog box, select Simba Apache Spark ODBC Connector and then click **Finish**. The Simba Apache Spark ODBC Connector Spark DSN Setup dialog box opens. The rest steps are the same as the step 6-21 of "*Creating a Data Source Name in Windows*" section.
6. To save your settings and close the Simba Spark ODBC Driver Configuration tool, click **OK**.



Note:

- a. Before you click OK and save the FileDSN, please confirm the test connection is successful. FileDSN could not be saved without a successful connection.
- b. If you do not want to save your password, do not change the "Password Options". If you want to save the password to FileDSN, choose "Current User Only" or "All Users of this Machine", your password will be saved in encrypted format in the FileDSN. You cannot remove the saved password from the dialog once it is saved. If you want to remove your password from your FileDSN, you can open the file and remove it manually.
- c. After the FileDSN is created. You may see an error "General error: Invalid file dsn", when you configure it from ODBC Data Source or use it in an application. This is due to a limitation of the Windows Driver Manager. If the FileDSN is too big, choose to use a "User DSN", "System DSN" or "DSN-less Connection" instead, refer to [Creating a Data Source Name in Windows](#) or [Configuring a DSN-less Connection in Windows](#).
- d. In some cases, the DM may silently truncate the FileDSN without throwing the error mentioned in (c), for example, if you have a long Auth_AccessToken saved in the FileDSN, the DM may pass a truncated Auth_AccessToken to the driver so that the driver may fail on connection.

Configuring a DSN-less Connection in Windows

Some client applications provide support for connecting to a data source using a connector without a Data Source Name (DSN). To configure a DSN-less connection, you can use a connection string or the Simba Spark ODBC Driver Configuration tool that is installed with the Simba Apache Spark ODBC Connector. Settings in a connection string apply only when you connect to Spark using that particular string, while settings in the connector configuration tool apply to every connection that uses the Simba Apache Spark ODBC Connector.

The following section explains how to use the connector configuration tool. For information about using connection strings, see [Using a Connection String](#).

 **Note:**

- Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.
- The drop-down lists in the connector configuration tool only display one option at a time. Use the scroll arrows on the right side of the drop-down list to view and select other options.

To configure a DSN-less connection using the connector configuration tool:

1. Choose one:

- Click Start  > All Programs > Simba Apache Spark ODBC Connector 2.9 > Driver Configuration.
- Or, click the arrow button at the bottom of the Start screen, and then click Simba Apache Spark ODBC Connector 2.9 > Driver Configuration.



Note: Make sure to select the Driver Configuration Tool that has the same bitness as the client application that you are using to connect to Spark.

2. If you are prompted for administrator permission to make modifications to the machine, click **OK**.



Note: You must have administrator access to the machine to run this application because it makes changes to the registry.

3. From the **Spark Server Type** drop-down list, select the appropriate server type for the version of Spark that you are running:

- If you are running Shark 0.8.1 or earlier, then select **SharkServer**.
- If you are running Shark 0.9, Spark 1.1 or later, then select **SparkThriftServer**.
- If you are connecting to Microsoft Synapse, then select **Synapse**.
- If you are connecting to Microsoft Fabric, then select **Fabric**.

4. Specify whether the connector uses the DataStax AOSS service when connecting to Spark:

- To connect to Spark without using the DataStax AOSS service, from the **Service Discovery Mode** drop-down list, select **No Service Discovery**.
- Or, to discover Spark services via the DataStax AOSS service, do the following:
 - From the **Service Discovery Mode** drop-down list, select **AOSS**.
 - If the AOSS endpoints require different authentication and SSL settings than the Spark service that you are connecting to, click **Service Discovery Options** and configure the

options in the AOSS Options dialog box as needed. For more information, see [Configuring AOSS Options in Windows](#).

5. In the Authentication area, configure authentication as needed. For more information, see [Configuring Authentication in Windows](#).

 **Note:** Shark Server does not support authentication. Most default configurations of Spark Thrift Server require User Name authentication. To verify the authentication mechanism that you need to use for your connection, check the configuration of your Hadoop / Spark distribution. For more information, see [Authentication Mechanisms](#).

6. Optionally, if the operations against Spark are to be done on behalf of a user that is different than the authenticated user for the connection, then in the **Delegation UID** field, type the name of the user to be delegated.

 **Note:** This option is applicable only when connecting to a Spark Thrift Server instance that supports this feature.

7. From the **Thrift Transport** drop-down list, select the transport protocol to use in the Thrift layer.

 **Note:** For information about how to determine which Thrift transport protocols your Spark server supports, see [Authentication Mechanisms](#).

8. If the Thrift Transport option is set to HTTP, then to configure HTTP options such as custom headers, click **HTTP Options**. For more information, see [Configuring HTTP Options in Windows](#).

9. To configure the connector to connect to Spark through a proxy server, click **Proxy Options**. For more information, see [Configuring a Proxy Connection in Windows](#).

10. To configure client-server verification over SSL, click **SSL Options**. For more information, see [Configuring SSL Verification in Windows](#).

 **Note:** If you selected User Name or Windows Azure HDInsight Emulator as the authentication mechanism, SSL is not available.

11. To configure advanced options, click **Advanced Options**. For more information, see [Configuring Advanced Options in Windows](#).

12. To configure server-side properties, click **Advanced Options** and then click **Server Side Properties**. For more information, see [Configuring Server-Side Properties in Windows](#).

13. To configure logging behavior for the connector, click **Logging Options**. For more information, see [Configuring Logging Options in Windows](#).

14. To save your settings and close the Simba Spark ODBC Driver Configuration tool, click **OK**.

Configuring AOSS Options in Windows

If you have configured the connector to discover Spark services via the DataStax AOSS service, and your AOSS endpoints require different authentication and SSL settings than the Spark service that you are connecting to, then you must also configure the connector to provide the necessary AOSS credentials.

The following instructions describe how to configure advanced options in a DSN and in the connector configuration tool. You can specify the connection settings described below in a DSN, in a connection string, or as connector-wide settings. Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

To configure AOSS options in Windows:

1. Choose one:
 - To access AOSS options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Service Discovery Options**.
 - Or, to access advanced options for a DSN-less connection, open the Simba Spark ODBC Driver Configuration tool, and then click **Service Discovery Options**.
2. To configure authentication, do the following:
 - a. From the **Status Endpoint Authentication** drop-down list, select **User Name and Password**.
 - b. In the **User Name** field, type the user name that you use to access the AOSS endpoints.
 - c. In the **Password** field, type the password corresponding to the user name you typed above.
 - d. To encrypt your credentials, click **Password Options** and then select one of the following:
 - If the credentials are used only by the current Windows user, select **Current User Only**.
 - Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.
3. To confirm your choice and close the Password Options dialog box, click **OK**.
3. To configure SSL encryption and identity verification, click **SSL Options** and then do the following:
 - a. Select the **Enable SSL** check box.
 - b. To allow authentication using self-signed certificates that have not been added to the list of trusted certificates, select the **Allow Self-signed Server Certificate** check box.
 - c. To allow the common name of a CA-issued SSL certificate to not match the host name of the AOSS endpoint, select the **Allow Common Name Host Name Mismatch** check box.
 - d. To specify the CA certificates that you want to use to verify the AOSS endpoint, do one of the following:
 - To verify the endpoint using the trusted CA certificates from a specific **.pem** file, specify the full path to the file in the **Trusted Certificates** field and clear the **Use System Trust Store** check box.
 - Or, to use the trusted CA certificates **.pem** file that is installed with the connector, leave the **Trusted Certificates** field empty, and clear the **Use System Trust Store** check box.

- Or, to use the Windows trust store, select the **Use System Trust Store** check box.

**Important:**

- If you are using the Windows trust store, make sure to import the trusted CA certificates into the trust store.
- If the trusted CA supports certificate revocation, select the **Check Certificate Revocation** check box.

- e. From the **Minimum TLS Version** drop-down list, select the minimum version of TLS to use when connecting to the endpoint.
- f. To save your SSL settings and close the SSL Options dialog box, click **OK**.

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

Configuring Authentication in Windows

Some Spark Thrift Server instances are configured to require authentication for access. To connect to a Spark server, you must configure the Simba Apache Spark ODBC Connector to use the authentication mechanism that matches the access requirements of the server and provides the necessary credentials.

For information about how to determine the type of authentication your Spark server requires, see [Authentication Mechanisms](#).

You can specify authentication settings in a DSN, in a connection string, or as connector-wide settings. Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

If cookie-based authentication is enabled in your Spark Server 2 database, you can specify a list of authentication cookies in the `HTTPAuthCookies` connection property. In this case, the connector authenticates the connection once based on the provided authentication credentials. It then uses the cookie generated by the server for each subsequent request in the same connection. For more information, see [HTTPAuthCookies](#).



Note: In Windows, the `HTTPAuthCookies` property must be set in a connection string.

Using No Authentication

When connecting to a Spark server of type Shark Server, you must use No Authentication. When you use No Authentication, Binary is the only Thrift transport protocol that is supported.

To configure a connection without authentication:

1. Choose one:
 - To access authentication options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, and then click **Configure**.

- Or, to access authentication options for a DSN-less connection, open the Simba Apache Spark ODBC Connector Configuration tool.

2. From the **Mechanism** drop-down list, select **No Authentication**.
3. If the Spark server is configured to use SSL, then click **SSL Options** to configure SSL for the connection. For more information, see [Configuring SSL Verification in Windows](#).
4. To save your settings and close the dialog box, click **OK**.

Using Kerberos

If the Use Only SSPI advanced option is disabled, then Kerberos must be installed and configured before you can use this authentication mechanism. For information about configuring Kerberos on your machine, see [Configuring Kerberos Authentication for Windows](#). For information about setting the Use Only SSPI advanced option, see [Configuring Advanced Options in Windows](#).



Note: This authentication mechanism is available only for Spark Thrift Server on non-HDInsight distributions.

To configure Kerberos authentication:

1. Choose one:
 - To access authentication options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, and then click **Configure**.
 - Or, to access authentication options for a DSN-less connection, open the Simba Apache Spark ODBC Connector Configuration tool.
2. From the **Mechanism** drop-down list, select **Kerberos**.
3. Choose one:
 - To use the default realm defined in your Kerberos setup, leave the **Realm** field empty.
 - Or, if your Kerberos setup does not define a default realm or if the realm of your Spark Thrift Server host is not the default, then, in the **Realm** field, type the Kerberos realm of the Spark Thrift Server.
4. In the **Host FQDN** field, type the fully qualified domain name of the Spark Thrift Server host.



Note: To use the Spark server host name as the fully qualified domain name for Kerberos authentication, in the **Host FQDN** field, type **_HOST**.

5. In the **Service Name** field, type the service name of the Spark server.
6. Optionally, if you are using MIT Kerberos and a Kerberos realm is specified in the **Realm** field, then choose one:
 - To have the Kerberos layer canonicalize the server's service principal name, leave the **Canonicalize Principal FQDN** check box selected.

- Or, to prevent the Kerberos layer from canonicalizing the server's service principal name, clear the **Canonicalize Principal FQDN** check box.

7. To allow the connector to pass your credentials directly to the server for use in authentication, select **Delegate Kerberos Credentials**.
8. From the **Thrift Transport** drop-down list, select the transport protocol to use in the Thrift layer.

Important:

When using this authentication mechanism, the Binary transport protocol is not supported.

9. If the Spark server is configured to use SSL, then click **SSL Options** to configure SSL for the connection. For more information, see [Configuring SSL Verification in Windows](#).
10. To save your settings and close the dialog box, click **OK**.

Using User Name

This authentication mechanism requires a user name but not a password. The user name labels the session, facilitating database tracking.



Note: This authentication mechanism is available only for Spark Thrift Server on non-HDInsight distributions. Most default configurations of Spark Thrift Server require User Name authentication.

To configure User Name authentication:

1. Choose one:
 - To access authentication options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, and then click **Configure**.
 - Or, to access authentication options for a DSN-less connection, open the Simba Apache Spark ODBC Connector Configuration tool.
2. From the **Mechanism** drop-down list, select **User Name**.
3. In the **User Name** field, type an appropriate user name for accessing the Spark server.
4. To save your settings and close the dialog box, click **OK**.

Using User Name And Password

This authentication mechanism requires a user name and a password.

To configure User Name And Password authentication:

1. Choose one:
 - To access authentication options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, and then click **Configure**.

- Or, to access authentication options for a DSN-less connection, open the Simba Apache Spark ODBC Connector Configuration tool.

2. From the **Mechanism** drop-down list, select **User Name And Password**.
3. In the **User Name** field, type an appropriate user name for accessing the Spark server.
4. In the **Password** field, type the password corresponding to the user name you typed above.
5. To encrypt your credentials, click **Password Options** and then select one of the following:
 - If the credentials are used only by the current Windows user, select **Current User Only**.
 - Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.To confirm your choice and close the Password Options dialog box, click **OK**.
6. From the **Thrift Transport** drop-down list, select the transport protocol to use in the Thrift layer.
7. If the Spark server is configured to use SSL, then click **SSL Options** to configure SSL for the connection. For more information, see [Configuring SSL Verification in Windows](#).
8. To save your settings and close the dialog box, click **OK**.

Using Windows Azure HDInsight Emulator

This authentication mechanism is available only for Spark Thrift Server instances running in Windows Azure HDInsight Emulator.

To configure a connection to a Spark server in Windows Azure HDInsight Emulator:

1. Choose one:
 - To access authentication options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, and then click **Configure**.
 - Or, to access authentication options for a DSN-less connection, open the Simba Spark ODBC Driver Configuration tool.
2. From the **Mechanism** drop-down list, select **Windows Azure HDInsight Emulator**.
3. In the **User Name** field, type an appropriate user name for accessing the Spark server.
4. In the **Password** field, type the password corresponding to the user name you specified above.
5. To encrypt your credentials, click **Password Options** and then select one of the following:
 - If the credentials are used only by the current Windows user, select **Current User Only**.
 - Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.

To confirm your choice and close the Password Options dialog box, click **OK**.

6. Click **HTTP Options**, and in the **HTTP Path** field, type the partial URL corresponding to the Spark server. Click **OK** to save your HTTP settings and close the dialog box.
7. To save your settings and close the dialog box, click **OK**.

Using Windows Azure HDInsight Service

This authentication mechanism is available only for Spark Thrift Server on HDInsight distributions.

To configure a connection to a Spark server in Windows Azure HDInsight Service:

1. Choose one:
 - To access authentication options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, and then click **Configure**.
 - Or, to access authentication options for a DSN-less connection, open the Simba Apache Spark ODBC Connector Configuration tool.
2. From the **Mechanism** drop-down list, select **Windows Azure HDInsight Service**.
3. In the **User Name** field, type an appropriate user name for accessing the Spark server.
4. In the **Password** field, type the password corresponding to the user name you typed above.
5. To encrypt your credentials, click **Password Options** and then select one of the following:
 - If the credentials are used only by the current Windows user, select **Current User Only**.
 - Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.

To confirm your choice and close the Password Options dialog box, click **OK**.

6. Click **HTTP Options**, and in the **HTTP Path** field, type the partial URL corresponding to the Spark server. Click **OK** to save your HTTP settings and close the dialog box.



Note: If necessary, you can create custom HTTP headers. For more information, see [Configuring HTTP Options in Windows](#).

7. Click **SSL Options** and configure SSL settings as needed. For more information, see [Configuring SSL Verification in Windows](#).
8. Click **OK** to save your SSL configuration and close the dialog box, and then click **OK** to save your authentication settings and close the dialog box.

Using OAuth 2.0

Four types of authentication work flow are available when using OAuth 2.0, token pass-through, client credentials, browser based authentication, or Azure Managed Identity

This authentication mechanism is available for Spark Thrift Server instances and instances only. When you use OAuth 2.0 authentication, HTTP is the only Thrift transport protocol available. Client credentials and browser based authentication work flow only works when SSL is enabled.

There is a discovery mode that enables the connector to auto-fill some endpoints or configurations. The endpoint discovery is enabled by default, you can disable it by setting `EnableOIDCDiscovery=0`. You can also pass the OIDC discovery endpoint by using `OIDCDiscoveryEndpoint`. The connector automatically discovers `OAuth2AuthorizationEndPoint` and `OAuth2TokenEndPoint`.

Token Pass-through

This authentication mechanism requires a valid OAuth 2.0 access token. Be aware that access tokens typically expire after a certain amount of time, after which you must either refresh the token or obtain a new one from the server. To obtain a new access token, see [Using OAuth 2.0](#).

To configure OAuth 2.0 token pass-through authentication:

1. Choose one:
 - To access authentication options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, and then click **Configure**.
 - Or, to access authentication options for a DSN-less connection, open the Simba Spark ODBC Driver Configuration tool.
2. From the **Mechanism** drop-down list, select **OAuth 2.0**.
3. Click **OAuth Options**, and then do the following:
 - a. From the **Authentication Flow** drop-down list, select **Token Passthrough**.
 - b. In the **Access Token** field, type your access token.
 - c. To save your settings and close the OAuth Options dialog box, click **OK**.
4. To save your settings and close the DSN Setup dialog box or the Driver Configuration tool, click **OK**.

Example connection string:

Providing a New Access Token

Once an access token expires, you can provide a new access token for the connector.



Note: When an access token expires, the connector returns a "SQLState 08006" error.

To obtain a new access token:

1. In the connection string, set the `Auth_AccessToken` property with a new access token.
2. Call the `SQLSetConnectAttr` function with `SQL_ATTR_CREDENTIALS` (122) as the attribute and the new connection string as the value. The connector will update the current connection string with the new access token.
3. Call the `SQLSetConnectAttr` function with `SQL_ATTR_REFRESH_CONNECTION` (123) as the attribute and `SQL_REFRESH_NOW` (-1) as the value. This signals the connector to update the



Note: Calling the `SQLGetConnectAttr` function with `SQL_ATTR_CREDENTIALS` (122) returns the entire connection string used during connection.

access token value.

4. Retry the previous ODBC API call. After obtaining the new access token, the open connection, statements, and cursors associated with it remain valid for use.

Client Credentials

This authentication mechanism requires SSL to be enabled.

You can use client secret or JWT assertion as the client credentials.

To configure OAuth 2.0 client credentials authentication using the client secret:

1. Choose one:
 - To access authentication options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, and then click **Configure**.
 - Or, to access authentication options for a DSN-less connection, open the Simba Spark ODBC Driver Configuration tool.
2. From the **Mechanism** drop-down list, select **OAuth 2.0**.
3. Click **OAuth Options**, and then do the following:
 - a. From the **Authentication Flow** drop-down list, select **Client Credentials**.
 - b. In the **Client ID** field, type your client ID.
 - c. In the **Client Secret** field, type your client secret.
 - d. Optionally, select **Encryption Options...** and choose the encryption password for **Current User Only** or **All Users of this Machine**. Then click **OK**.
 - e. Optionally, select the **Ignore SQL_DRIVER_NOPROMPT** check box. When the application is making a SQLDriverConnect call with a SQL_DRIVER_NOPROMPT flag, this option displays the web browser used to complete the browser based authentication flow.
 - f. To save your settings and close the OAuth Options dialog box, click **OK**.
4. To save your settings and close the DSN Setup dialog box or the Driver Configuration tool, click **OK**.

To configure OAuth 2.0 client credentials authentication using the JWT assertion:

1. Choose one:
 - To access authentication options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, and then click **Configure**.
 - Or, to access authentication options for a DSN-less connection, open the Simba Spark ODBC Driver Configuration tool.
2. From the **Mechanism** drop-down list, select **OAuth 2.0**.
3. Click **OAuth Options**, and then do the following:
 - a. From the **Authentication Flow** drop-down list, select **Client Credentials**.
 - b. Select the **Use JWT Assertion** check box.

- c. In the **Client ID** field, type your client ID.
- d. In the **JWT Key Identifier** field, type your key identifier.
- e. In the **JWT Private Key Path** field, select your private key pem file.
- f. In the **JWT Private Key Password** field, type your passphrase, if your private key is encrypted.
- g. Optionally, click **JWT Private Key Encryption Options** and select the encryption password for **Current User Only** or **All Users** of this Machine. Click **OK**.
- h. In the **OIDC Discovery Endpoint** field, type your discovery endpoint.
- i. To save your settings and close the OAuth Options dialog box, click **OK**.

4. To save your settings and close the DSN Setup dialog box or the Driver Configuration tool, click **OK**.

Browser Based

This authentication mechanism requires SSL to be enabled.

To configure OAuth 2.0 browser based authentication:

1. Choose one:
 - To access authentication options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, and then click **Configure**.
 - Or, to access authentication options for a DSN-less connection, open the SimbaSpark ODBC Driver Configuration tool.
2. From the **Mechanism** drop-down list, select **OAuth 2.0**.
3. Click **OAuth Options**, and then do the following:
 - a. From the **Authentication Flow** drop-down list, select **Browser Based Authorization Code**.
 - b. Optionally, select the **Ignore SQL_DRIVER_NOPROMPT** check box. When the application is making a SQLDriverConnect call with a SQL_DRIVER_NOPROMPT flag, this option displays the web browser used to complete the browser based authentication flow.
 - c. To save your settings and close the OAuth Options dialog box, click **OK**.
4. To save your settings and close the DSN Setup dialog box or the Driver Configuration tool, click **OK**.



Note: When the browser based authentication flow completes, the access token and refresh token are saved in the token cache and the connector does not need to authenticate again. For more information, see [Enable Token Cache](#).

Azure Managed Identity

This authentication mechanism requires SSL to be enabled.

To configure Azure Managed Identity based authentication:

1. Choose one:
 - To access authentication options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, and then click **Configure**.
 - Or, to access authentication options for a DSN-less connection, open the SimbaSpark ODBC Driver Configuration tool.
2. From the **Mechanism** drop-down list, select **OAuth 2.0**.
3. Click **OAuth Options**, and then do the following:
 - a. From the **Authentication Flow** drop-down list, select **Azure Managed Identity**.
 - b. Optionally, in the **Client ID** field, type the user-assigned managed identity.
 - c. Optionally, in the **Azure Workspace Resource ID**, type your Resource ID.
 - d. To save your settings and close the OAuth Options dialog box, click **OK**.
4. To save your settings and close the DSN Setup dialog box or the Driver Configuration tool, click **OK**.

Using an API Signing Key in Windows

This authentication mechanism requires you to have credentials stored in an OCI configuration file.

The following instructions describe how to configure advanced options in a DSN and in the connector configuration tool. You can specify the connection settings described below in a DSN, in a connection string, or as connector-wide settings. Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

To configure authentication using an API signing key in Windows:

1. Choose one:
 - To configure authentication for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **OCI Options**.
 - Or, to configure authentication for a DSN-less connection, open the Simba Spark ODBC Driver Configuration tool, and then click **OCI Options**.
2. Set the **OCI Config File** field to the absolute path to the OCI configuration file to use for the connection.
3. Optionally, set the **OCI Profile** field to the name of the OCI profile to use for the connection. If no profile is specified, the connector attempts to use the profile named **DEFAULT**.
4. If the OCI profile or configuration file includes a password, do the following:
 - a. In the **Key File Password** field, type the password.
 - b. To encrypt your credentials, click **Password Options** and then select one of the following:
 - If the credentials are used only by the current Windows user, select **Current User Only**.
 - Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.

To confirm your choice and close the Password Options dialog box, click **OK**.

5. To save your settings and close the OCI Options dialog box, click **OK**.

When you initiate a connection using an API signing key, the connector uses the following behavior in case of errors:

1. If no configuration file is specified, the connector first attempts to locate the configuration file in the default location:
 - For Windows, the default location is: %HOMEDRIVE%%HOMEPATH%\oci\config
 - For non-Windows platforms, the default location is: ~/.oci/config
2. If the configuration file cannot be located in the default location and the OCI_CLI_CONFIG_FILE environment variable is specified, the connector next attempts to locate the configuration file using the value of the environment variable.
3. If the configuration file cannot be located, or the profile cannot be opened, the connector falls back to token-based authentication. For more information, see [Using Token-based Authentication in Windows](#).
4. If an error occurs when using the credentials from the profile, the connector returns an error. In this case, the connector does not fall back to token-based authentication.

Using Token-based Authentication in Windows

Token-based authentication is used to interactively authenticate the connection for a single session.

When you initiate a connection using token-based authentication, the connector attempts to open a web browser. You may be prompted to type your credentials in the browser.

The following instructions describe how to configure advanced options in a DSN and in the connector configuration tool. You can specify the connection settings described below in a DSN, in a connection string, or as connector-wide settings. Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

To configure authentication using token-based authentication in Windows:

1. Choose one:
 - To configure authentication for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **OCI Options**.
 - Or, to configure authentication for a DSN-less connection, open the Simba Spark ODBC Driver Configuration tool, and then click **OCI Options**.
2. Set the **OCI Config File** field to a path that does not contain an OCI configuration file.
3. To display a web browser used to complete the token-based authentication flow even when SQL_DRIVER_NOPROMPT is enabled, select the **Ignore SQL_DRIVER_NOPROMPT** check box.
4. To save your settings and close the OCI Options dialog box, click **OK**.

Configuring Advanced Options in Windows

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

The following instructions describe how to configure advanced options in a DSN and in the connector configuration tool. You can specify the connection settings described below in a DSN, in a connection string, or as connector-wide settings. Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

To configure advanced options in Windows:

1. Choose one:
 - To access advanced options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Advanced Options**.
 - Or, to access advanced options for a DSN-less connection, open the Simba Apache Spark ODBC Connector Configuration tool, and then click **Advanced Options**.
2. To disable the SQL Connector feature, select the **Use Native Query** check box.

 **Important:**

- When this option is enabled, the connector cannot execute parameterized queries.
- By default, the connector applies transformations to the queries emitted by an application to convert the queries into an equivalent form in HiveQL. If the application is Spark-aware and already emits HiveQL, then turning off the translation avoids the additional overhead of query transformation.

3. To defer query execution to SQLExecute, select the **Fast SQLPrepare** check box.
4. To allow connector-wide configurations to take precedence over connection and DSN settings, select the **Driver Config Take Precedence** check box.
5. To use the asynchronous version of the API call against Spark for executing a query, select the **Use Async Exec** check box.
6. To retrieve table names from the database by using the SHOW TABLES query, select the **Get Tables With Query** check box.

 **Note:** This option is applicable only when connecting to Spark Thrift Server.

7. To enable the connector to return SQL_WVARCHAR instead of SQL_VARCHAR for STRING and VARCHAR columns, and SQL_WCHAR instead of SQL_CHAR for CHAR columns, select the **Unicode SQL Character Types** check box.
8. To enable the connector to return the spark_system table for catalog function calls such as SQLTables and SQLColumns, select the **Show System Table** check box.
9. To specify which mechanism the connector uses by default to handle Kerberos authentication, do one of the following:

- To use the SSPI plugin by default, select the **Use Only SSPI** check box.
- To use MIT Kerberos by default and only use the SSPI plugin if the GSSAPI library is not available, clear the **Use Only SSPI** check box.

10. To enable the connector to automatically open a new session when the existing session is no longer valid, select the **Invalid Session Auto Recover** check box.

 **Note:** This option is applicable only when connecting to Spark Thrift Server.

11. To have the connector automatically attempt to reconnect to the server if communications are lost, select **AutoReconnect**.

12. To enable the connector to perform a query translation for the CREATE TABLE AS SELECT (CTAS) syntax, select **Enable Transaction For CTAS**.

13. In the **Rows Fetched Per Block** field, type the number of rows to be fetched per block.

14. In the **Max Bytes Per Fetch Request** field, type the maximum number of bytes to be fetched.

 **Note:** This option is applicable only when connecting to a server that supports result set data serialized in arrow format. The value must be specified in one of the following:

- B (bytes)
- KB (kilobytes)
- MB (megabytes)
- GB (gigabytes)

By default, the file size is in B (bytes).

15. In the **Default String Column Length** field, type the maximum data length for STRING columns.

16. In the **Binary Column Length** field, type the maximum data length for BINARY columns.

17. In the **Decimal Column Scale** field, type the maximum number of digits to the right of the decimal point for numeric data types.

18. In the **Async Exec Poll Interval** field, type the time in milliseconds between each poll for the query execution status.

 **Note:** This option is applicable only to HDInsight clusters.

19. In the **Socket Timeout** field, type the number of seconds that an operation can remain idle before it is closed.

 **Note:** This option is applicable only when asynchronous query execution is being used against Spark Thrift Server instances.

20. In the **Query Timeout Override** field, type the number of seconds that a query can run before it is timed out.



Note: When the value passed is an empty string, the connector does not attempt to override the `SQL_ATTR_QUERY_TIMEOUT` attribute.

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

Configuring a Proxy Connection in Windows

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

To configure a proxy server connection in Windows:

1. To access proxy server options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Proxy Options**.
2. Select the **Use Proxy** check box.
3. In the **Proxy Host** field, type the host name or IP address of the proxy server.
4. In the **Proxy Port** field, type the number of the TCP port that the proxy server uses to listen for client connections.
5. From the **Auth Type** drop-down list, choose one:
 - a. **Basic Auth:**
 - i. In the **Proxy Username** field, type your user name for accessing the proxy server.
 - ii. In the **Proxy Password** field, type the password corresponding to the user name.
 - b. **Kerberos:**
 - i. In the **Realm** field, the connector takes the default realm of the system.
 - ii. In the **Host FQDN** field, the connector takes the host FQDN of the ProxyHost.
 - iii. In the **Service Name** field, the connector takes the default HTTP.
 - c. **No Auth:** The connector does not authenticate.
6. To encrypt your credentials, click **Password Options** and then select one of the following:
 - If the credentials are used only by the current Windows user, select **Current User Only**.
 - Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.

To confirm your choice and close the Password Options dialog box, click **OK**.

7. In the **Proxy Ignore List** field, type the list of hosts or domains that do not use a proxy.
8. To save your settings and close the HTTP Proxy Options dialog box, click **OK**.

Configuring HTTP Options in Windows

You can configure options such as custom headers when using the HTTP transport protocol in the Thrift layer. For information about how to determine if your Spark server supports the HTTP transport protocol, see [Authentication Mechanisms](#).

The following instructions describe how to configure HTTP options in a DSN in the connector configuration tool. You can specify the connection settings described below in a DSN, in a connection string, or as connector-wide settings. Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

To configure HTTP options in Windows:

1. To access HTTP options, click **HTTP Options**.



Note: The HTTP options are available only when the Transport option is set to HTTP.

2. In the **HTTP Path** field, type the partial URL corresponding to the Spark server.
3. To create a custom HTTP header, click **Add**, then type appropriate values in the **Key** and **Value** fields, and then click **OK**.
4. To edit a custom HTTP header, select the header from the list, then click **Edit**, then update the **Key** and **Value** fields as needed, and then click **OK**.
5. To delete a custom HTTP header, select the header from the list, and then click **Remove**. In the confirmation dialog box, click **Yes**.
6. To save your settings and close the HTTP Properties dialog box, click **OK**.

Configuring SSL Verification in Windows

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

The following instructions describe how to configure SSL in a DSN and in the connector configuration tool. You can specify the connection settings described below in a DSN, in a connection string, or as connector-wide settings. 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 you selected User Name or Windows Azure HDInsight Emulator as the authentication mechanism, SSL is not available.

To configure SSL verification in Windows:

1. Choose one:
 - To access SSL options for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **SSL Options**.
 - Or, to access advanced options for a DSN-less connection, open the Simba Apache Spark ODBC Connector Configuration tool, and then click **SSL Options**.

2. Select the **Enable SSL** check box.
3. To allow authentication using self-signed certificates that have not been added to the list of trusted certificates, select the **Allow Self-signed Server Certificate** check box.
4. To allow the common name of a CA-issued SSL certificate to not match the host name of the Spark server, select the **Allow Common Name Host Name Mismatch** check box.
5. To specify the CA certificates that you want to use to verify the server, do one of the following:
 - To verify the server using the trusted CA certificates from a specific .pem file, specify the full path to the file in the **Trusted Certificates** field and clear the **Use System Trust Store** check box.
 - Or, to use the trusted CA certificates .pem file that is installed with the connector , leave the **Trusted Certificates** field empty, and clear the **Use System Trust Store** check box.
 - Or, to use the Windows trust store, select the **Use System Trust Store** check box.

 **Important:**

- If you are using the Windows trust store, make sure to import the trusted CA certificates into the trust store.
- If the trusted CA supports certificate revocation, select the **Check Certificate Revocation** check box.

6. To allow authentication, when the certificate's revocation status is undetermined, select the **Accept Undetermined Revocation** checkbox.
7. From the **Minimum TLS Version** drop-down list, select the minimum version of TLS to use when connecting to your data store.
8. To configure two-way SSL verification, select the **Two-Way SSL** check box and then do the following:
 - a. In the **Client Certificate File** field, specify the full path of the PEM file containing the client's certificate.
 - b. In the **Client Private Key File** field, specify the full path of the file containing the client's private key.
 - c. If the private key file is protected with a password, type the password in the **Client Private Key Password** field.

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

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

To confirm your choice and close the Password Options dialog box, click **OK**.

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

Configuring Server-Side Properties in Windows

You can use the connector to apply configuration properties to the Spark server.

The following instructions describe how to configure server-side properties in a DSN and in the connector configuration tool. You can specify the connection settings described below in a DSN, in a connection string, or as connector-wide settings. Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

To configure server-side properties in Windows:

1. Choose one:
 - To configure server-side properties for a DSN, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, then click **Advanced Options**, and then click **Server Side Properties**.
 - Or, to configure server-side properties for a DSN-less connection, open the Simba Apache Spark ODBC Connector Configuration tool, then click **Advanced Options**, and then click **Server Side Properties**.
2. To create a server-side property, click **Add**, then type appropriate values in the **Key** and **Value** fields, and then click **OK**.

Note: For a list of all Hadoop and Spark server-side properties that your implementation supports, type `set -v` at the Spark CLI command line. You can also execute the `set -v` query after connecting using the connector.

3. To edit a server-side property, select the property from the list, then click **Edit**, then update the **Key** and **Value** fields as needed, and then click **OK**.
4. To delete a server-side property, select the property from the list, and then click **Remove**. In the confirmation dialog box, click **Yes**.
5. To configure the connector to convert server-side property key names to all lower-case characters, select the **Convert Key Name To Lower Case** check box.
6. To change the method that the connector uses to apply server-side properties, do one of the following:
 - To configure the connector to apply each server-side property by executing a query when opening a session to the Spark server, select the **Apply Server Side Properties With Queries** check box.
 - Or, to configure the connector to use a more efficient method for applying server-side properties that does not involve additional network round-tripping, clear the **Apply Server Side Properties With Queries** check box.

www.insightsoftware.com

i **Note:** The more efficient method is not available for Shark Server, and it might not be compatible with some Spark Thrift Server builds. If the server-side properties do not take effect when the check box is clear, then select the check box.

7. To save your settings and close the Server Side Properties dialog box, click **OK**.

Configuring Logging Options in Windows

To help troubleshoot issues, you can enable logging. In addition to functionality provided in the Simba Simba Apache Spark 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 Apache Spark ODBC Connector, so make sure to disable the feature after you are done using it. To configure logging for the current connection, see [Configuring Logging for the Current Connection](#).

To enable connector-wide logging in Windows:

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

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

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

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

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 Apache Spark ODBC Connector produces the following log files at the location you specify in the Log Path field:

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

To disable connector logging in Windows:

1. Open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Logging Options**.
2. From the **Log Level** drop-down list, select **LOG_OFF**.
3. Click **OK**.
4. Restart your ODBC application to make sure that the new settings take effect.

Configuring Logging for the Current Connection

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



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

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



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

To add logging configurations to a DSN in Windows:

1. 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.

Configuring Kerberos Authentication for Windows Active Directory

The Simba Apache Spark ODBC Connector supports Active Directory Kerberos in Windows. There are two prerequisites for using Active Directory Kerberos in Windows:

- MIT Kerberos is not installed on the client Windows machine.
- The MIT Kerberos Hadoop realm has been configured to trust the Active Directory realm so that users in the Active Directory realm can access services in the MIT Kerberos Hadoop realm.

MIT Kerberos

Downloading and Installing MIT Kerberos for Windows 4.0.1

For information about Kerberos and download links for the installer, see the MIT Kerberos website: <http://web.mit.edu/kerberos/>.

To download and install MIT Kerberos for Windows 4.0.1:

1. Download the appropriate Kerberos installer:

- For a 64-bit machine, use the following download link from the MIT Kerberos website:
<http://web.mit.edu/kerberos/dist/kfw/4.0/kfw-4.0.1-amd64.msi>.
- For a 32-bit machine, use the following download link from the MIT Kerberos website:
<http://web.mit.edu/kerberos/dist/kfw/4.0/kfw-4.0.1-i386.msi>.



Note: The 64-bit installer includes both 32-bit and 64-bit libraries. The 32-bit installer includes 32-bit libraries only.

2. To run the installer, double-click the `.msi` file that you downloaded above.

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

4. When the installation completes, click **Finish**.

Setting Up the Kerberos Configuration File

Settings for Kerberos are specified through a configuration file. You can set up the configuration file as an `.ini` file in the default location, which is the `C:\ProgramData\MIT\Kerberos5` directory, or as a `.conf` file in a custom location.

Normally, the `C:\ProgramData\MIT\Kerberos5` directory is hidden. For information about viewing and using this hidden directory, refer to Microsoft Windows documentation.



Note: For more information on configuring Kerberos, refer to the MIT Kerberos documentation.

To set up the Kerberos configuration file in the default location:

1. Obtain a `krb5.conf` configuration file. You can obtain this file from your Kerberos administrator, or from the `/etc/krb5.conf` folder on the machine that is hosting the Spark Thrift Server instance.
2. Rename the configuration file from `krb5.conf` to `krb5.ini`.
3. Copy the `krb5.ini` file to the `C:\ProgramData\MIT\Kerberos5` directory and overwrite the empty sample file.

To set up the Kerberos configuration file in a custom location:

1. Obtain a `krb5.conf` configuration file. You can obtain this file from your Kerberos administrator, or from the `/etc/krb5.conf` folder on the machine that is hosting the Spark Thrift Server instance.
2. Place the `krb5.conf` file in an accessible directory and make note of the full path name.
3. Open the System window:
 - Click **Start** , then right-click **Computer**, and then click **Properties**.
 - Or right-click **This PC** on the Start screen, and then click **Properties**.
4. Click **Advanced System Settings**.

5. In the System Properties dialog box, click the **Advanced** tab and then click **Environment Variables**.
6. In the Environment Variables dialog box, under the System Variables list, click **New**.
7. In the New System Variable dialog box, in the **Variable Name** field, type **KRB5_CONFIG**.
8. In the **Variable Value** field, type the full path to the `krb5.conf` file.
9. Click **OK** to save the new variable.
10. Make sure that the variable is listed in the System Variables list.
11. Click **OK** to close the Environment Variables dialog box, and then click **OK** to close the System Properties dialog box.

Setting Up the Kerberos Credential Cache File

Kerberos uses a credential cache to store and manage credentials.

To set up the Kerberos credential cache file:

1. Create a directory where you want to save the Kerberos credential cache file. For example, create a directory named `C:\temp`.
2. Open the System window:
 - Click **Start** , then right-click **Computer**, and then click **Properties**.
 - Or right-click **This PC** on the Start screen, and then click **Properties**.
3. Click **Advanced System Settings**.
4. In the System Properties dialog box, click the **Advanced** tab and then click **Environment Variables**.
5. In the Environment Variables dialog box, under the System Variables list, click **New**.
6. In the New System Variable dialog box, in the **Variable Name** field, type **KRB5CCNAME**.
7. In the **Variable Value** field, type the path to the folder you created above, and then append the file name `krb5cache`. For example, if you created the folder `C:\temp`, then type `C:\temp\krb5cache`.



Note: `krb5cache` is a file (not a directory) that is managed by the Kerberos software, and it should not be created by the user. If you receive a permission error when you first use Kerberos, make sure that the `krb5cache` file does not already exist as a file or a directory.

8. Click **OK** to save the new variable.
9. Make sure that the variable appears in the System Variables list.
10. Click **OK** to close the Environment Variables dialog box, and then click **OK** to close the System Properties dialog box.
11. To make sure that Kerberos uses the new settings, restart your machine.

Obtaining a Ticket for a Kerberos Principal

A principal refers to a user or service that can authenticate to Kerberos. To authenticate to Kerberos, a principal must obtain a ticket by using a password or a keytab file. You can specify a keytab file to use, or use the default keytab file of your Kerberos configuration.

To obtain a ticket for a Kerberos principal using a password:

1. Open MIT Kerberos Ticket Manager.
2. In MIT Kerberos Ticket Manager, click **Get Ticket**.
3. In the Get Ticket dialog box, type your principal name and password, and then click **OK**.

If the authentication succeeds, then your ticket information appears in MIT Kerberos Ticket Manager.

To obtain a ticket for a Kerberos principal using a keytab file:

- a. Open a command prompt:

- Click **Start** , then click **All Programs**, then click **Accessories**, and then click **Command Prompt**.
- Click the arrow button at the bottom of the Start screen, then find the Windows System program group, and then click **Command Prompt**.

- b. In the Command Prompt, type a command using the following syntax:

```
kinit -k -t [KeytabPath] [Principal]
```

[KeytabPath] is the full path to the keytab file. For example:

C:\mykeytabs\myUser.keytab.

[Principal] is the Kerberos user principal to use for authentication. For example:

myUser@EXAMPLE.COM.

- c. If the cache location KRB5CCNAME is not set or used, then use the **-c** option of the **kinit** command to specify the location of the credential cache. In the command, the **-c** argument must appear last. For example:

```
kinit -k -t C:\mykeytabs\myUser.keytab myUser@EXAMPLE.COM -c
C:\ProgramData\MIT\krbcache
```

Krbcache is the Kerberos cache file, not a directory.

To obtain a ticket for a Kerberos principal using the default keytab file:



Note: For information about configuring a default keytab file for your Kerberos configuration, refer to the MIT Kerberos documentation.

1. Open a command prompt:

- Click Start , then click All Programs, then click Accessories, and then click Command Prompt.
- Click the arrow button at the bottom of the Start screen, then find the Windows System program group, and then click Command Prompt.

2. In the Command Prompt, type a command using the following syntax:

```
kinit -k [principal]
```

[principal] is the Kerberos user principal to use for authentication. For example:

MyUser@EXAMPLE.COM.

3. If the cache location KRB5CCNAME is not set or used, then use the -c option of the kinit command to specify the location of the credential cache. In the command, the -c argument must appear last. For example:

```
kinit -k -t C:\mykeytabs\myUser.keytab myUser@EXAMPLE.COM -c C:\ProgramData\MIT\krbcache
```

Krbcache is the Kerberos cache file, not a directory.

Verifying the Connector Version Number in Windows

If you need to verify the version of the Simba Apache Spark 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 Spark.

2. Click the **Drivers** tab and then find the Simba Apache Spark 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

The Simba Apache Spark ODBC Connector supports Apache Spark versions 3.1.2 through 3.5.0.

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

- One of the following macOS versions:
 - macOS 12 (Universal Binary - Intel and ARM support)
 - macOS 13 (Universal Binary - Intel and ARM support)
 - macOS 14 (Universal Binary - Intel and ARM support)
 - macOS 15 (Universal Binary-Intel and ARM support)
- 100MB of available disk space
- One of the following ODBC driver managers installed:
 - iODBC 3.52.9 or later
 - unixODBC 2.3.6 or later

 **Note:** The recommended unixODBC version is 2.3.12 or later when the driver is used by applications that require a scrollable cursor.

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 Apache Spark ODBC Connector is available for macOS as a `.dmg` file named `SimbaSparkODBC.dmg`. The connector supports 64-bit intel and arm architecture.

To install the Simba Apache Spark ODBC Connector in macOS:

1. Double-click **SimbaSparkODBC.dmg** to mount the disk image.
2. Double-click **SimbaSpark 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/spark` 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/spark/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 Apache Spark 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.sparkodbc
```

The command returns information about the Simba Apache Spark 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.

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

Linux System Requirements

The Simba Apache Spark ODBC Connector supports Apache Spark versions 3.1.2 through 3.5.0.

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

- One of the following distributions:
 - Debian 11
 - Red Hat® Enterprise Linux® (RHEL) 8, 8ARM, 9, 9ARM, 10, 10ARM, or AL2023 ARM
 - SUSE Linux Enterprise Server (SLES) 15
 - Ubuntu 22.04 or 24.04
- 150 MB of available disk space
- One of the following ODBC driver managers installed:
 - iODBC 3.52.9 or later
 - unixODBC 2.2.14 or later



Note: The recommended unixODBC version is 2.3.12 or later when the driver is used by applications that require a scrollable cursor.

- The MIT kerberos libraries installed.
- All of the following `libsasl` libraries installed:
 - `cyrus-sasl-2.1.22-7 or later`
 - `cyrus-sasl-gssapi-2.1.22-7 or later`
 - `cyrus-sasl-plain-2.1.22-7 or later`
 - `libsasl2-modules-gssapi-mit 2.1.28`



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

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*.



Important: The RPM file package is meant to be used on machines running RHEL, CentOS, SUSE, or Oracle Linux only.

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:

- `simbasparkODBC-32bit-[Version]-[Release].i686.rpm` for the 32-bit connector
- `simbasparkODBC-[Version]-[Release].x86_64.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.

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

To install the Simba Apache Spark ODBC Connector using the RPM File:

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

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

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

```
zypper install [RPMFileName]
```

The Simba Apache Spark ODBC Connector files are installed in the `/opt/simba/spark` directory.



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

4. If you received a license file through email, then copy the license file into the `/opt/simba/spark/lib/32` or `/opt/simba/spark/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 Apache Spark ODBC Connector is available as a tarball package named `SimbaSparkODBC- [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 Apache Spark ODBC Connector files are installed in the `/opt/simba/spark` directory.

3. If you received a license file through email, then copy the license file into the `/opt/simba/spark/lib/32` or `/opt/simba/spark/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 on Debian

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



Important: The Debian package is meant to be used on machines running Debian or Ubuntu only.

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

- `simbaspark_[Version]-[Release]_i386.deb` for the 32-bit connector
- `simbaspark_[Version]-[Release]_amd64.deb` for the 64-bit connector

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

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

To install the Simba Apache Spark ODBC Connector on Debian:

1. Log in as the root user, and then navigate to the folder containing the Debian package for the connector.
2. Double-click `simbaspark_[Version]-[Release]_i386.deb` or `simbaspark_[Version]-[Release]_amd64.deb`.
3. Follow the instructions in the installer to complete the installation process.

The Simba Apache Spark ODBC Connector files are installed in the `/opt/simba/spark` directory.



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

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

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

Verifying the Connector Version Number in Linux

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

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

- Depending on your package manager, at the command prompt, run one of the following commands:

- `yum list | grep SimbaSparkODBC`

- `rpm -qa | grep SimbaSparkODBC`

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

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

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

AIX Connector

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

AIX System Requirements

The Simba Apache Spark ODBC Connector supports Apache Spark versions 2.4 through 3.0.

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

- IBM AIX 5.3, 6.1, or 7.1
- 150 MB of available disk space
- One of the following ODBC driver managers installed:
 - iODBC 3.52.9 or later
 - unixODBC 2.2.14 or later



Note: The recommended unixODBC version is 2.3.12 or later when the driver is used by applications that require a scrollable cursor.

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

Installing the Connector on AIX

On 64-bit editions of AIX, 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 the version of the connector that matches the bitness of the client application:

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

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

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

To install the Simba Apache Spark ODBC Connector on AIX:

1. Log in as the root user, and then navigate to the folder containing the RPM package for the connector.
2. Run the following command from the command line, where `[RPMFileName]` is the file name of the RPM package:

```
rpm --install [RPMFileName]
```

The Simba Apache Spark ODBC Connector files are installed in the `/opt/simba/spark` directory.

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

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

Verifying the Connector Version Number on AIX

If you need to verify the version of the Simba Apache Spark ODBC Connector that is installed on your AIX machine, you can query the version number through the command-line interface.

To verify the connector version number on AIX:

- At the command prompt, run the following command:

```
rpm -qa | grep SimbaSparkODBC
```

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

Solaris Connector

Solaris System Requirements

The Simba Apache Spark ODBC Connector supports Apache Spark versions 2.4 through 3.0.

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

- Solaris 10 or 11 (SPARC and SPARC64 editions only)
- 150 MB of available disk space
- One of the following ODBC driver managers installed:
 - iODBC 3.52.9 or later
 - unixODBC 2.2.14 or later



Note: The recommended unixODBC version is 2.3.12 or later when the driver is used by applications that require a scrollable cursor.

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

Installing the Connector on Solaris

The Simba Apache Spark ODBC Connector is available for Solaris as a tarball package named `Simba Apache Spark ODBC Connector_Solaris-_gcc_[Version].[Release]_Solaris.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 sparc64 editions of Solaris, you can execute both sparc and sparc64 applications. However, sparc64 applications must use 64-bit connectors, and sparc applications must use 32-bit connectors. Make sure that you use the version of the connector that matches the bitness of the client application. You can install both versions of the connector on the same machine.

To install the Simba Apache Spark ODBC Connector on Solaris:

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 Apache Spark ODBC Connector files are installed in the `/opt/simba/spark/` directory.

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

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

Verifying the Connector Version Number on Solaris

If you need to verify the version of the Simba Apache Spark ODBC Connector that is installed on your Solaris machine, you can query the version number through the command-line interface.

To verify the connector version number on Solaris:

- At the command prompt, run the following command:

```
rpm -qa | grep SimbaSparkODBC
```

The command returns information about the Simba Apache Spark 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 Apache Spark 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.sparkodbc.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 `SIMBASPARKINI` to the full path and file name of the `simba.sparkodbc.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 `SIMBASPARKINI` to the full path and file name of the `simba.sparkodbc.ini` file.

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

For unixODBC:

```
export ODBCINI=/usr/local/odbc/odbc.ini
export ODBC SYSINI=/usr/local/odbc
export SIMBASPARKINI=/etc/simba.sparkodbc.ini
```

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

1. If the `SIMBASPARKINI` 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.sparkodbc.ini`.
3. The connector searches the current working directory of the application for a file named `simba.sparkodbc.ini`.

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

Configuring ODBC Connections in Non-Windows Machine

The following sections describe how to configure ODBC connections when using the Simba Apache Spark 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 DataStax AOSS Service Discovery on a Non-Windows Machine](#)
- [Configuring Authentication on a Non-Windows Machine](#)
- [Configuring SSL Verification in a Non-Windows Machine](#)
- [Configuring Server-Side Properties on a Non-Windows Machine](#)
- [Configuring Logging Options in a Non-Windows Machine](#)
- [Setting Connector-Wide Configuration Options on a Non-Windows Machine](#)
- [Testing the Connection in Non-Windows Machine](#)

Creating a Data Source Name on a Non-Windows Machine

Typically, after installing the Simba Apache Spark ODBC Connector, you need to create a Data Source Name (DSN). A DSN is a data structure that stores connection information so that it can be used by the connector to connect to Spark.

You can specify connection settings in a DSN (in the `odbc.ini` file), in a connection string, or as connector-wide settings (in the `simba.sparkodbc.ini` file). Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

The following instructions describe how to create a DSN by specifying connection settings in the `odbc.ini` file. 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.

For information about specifying settings in a connection string, see [Configuring a DSN-less Connection in a Non-Windows Machine](#) and [Using a Connection String](#). For information about connector-wide settings, see [Setting Connector-Wide Configuration Options on a Non-Windows Machine](#).

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 Apache Spark ODBC Connector

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

`[ODBC Data Sources]`

Sample DSN=Simba Apache Spark 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/spark/lib/libsparkodbc_sb64-universal.dylib`

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

`Driver=/opt/simba/spark/lib/32/libsparkodbc_sb32.so`

- b. Set the `SparkServerType` property to one of the following values:

- If you are running Shark 0.8.1 or earlier, set the property to 1.
- If you are running Shark 0.9 or Spark 1.1 or later, set the property to 3.

For example:

`SparkServerType=3`

- c. Specify whether the connector uses the DataStax AOSS service when connecting to Spark, and provide the necessary connection information:

- To connect to Spark without using the DataStax AOSS service, do the following:
 - Set the `ServiceDiscoveryMode` property to `No Service Discovery`.
 - Set the `Host` property to the IP address or host name of the Spark server.
 - Set the `Port` property to the number of the TCP port that the Spark server uses to listen for client connections.

For example:

ServiceDiscoveryMode=No Service Discovery

Host=192.168.222.160

Port=10000

- Or, to discover Spark services via the DataStax AOSS service, set properties as described in [Configuring DataStax AOSS Service Discovery on a Non-Windows Machine](#).
- d. If authentication is required to access the Spark server, then specify the authentication mechanism and your credentials. For more information, see [Configuring Authentication on a Non-Windows Machine](#).
- e. If you want to connect to the server through SSL, then enable SSL and specify the certificate information. For more information, see [Configuring SSL Verification in a Non-Windows Machine](#).



Note: If the `AuthMech` property is set to 2 or 5, SSL is not available.

- f. If you want to configure server-side properties, then set them as key-value pairs using a special syntax. For more information, see [Configuring Server-Side Properties on a Non-Windows Machine](#).
- g. Optionally, set additional key-value pairs as needed to specify other optional connection settings. For detailed information about all the configuration options supported by the Simba Apache Spark ODBC Connector, see [Connector Configuration Options](#).

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



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

For example, the following is an `odbc.ini` configuration file for macOS containing a DSN that connects to a Spark Thrift Server instance and authenticates the connection using a user name and password:

[ODBC Data Sources]

Sample DSN=Simba Apache Spark ODBC Connector

[Sample DSN]

Driver=/Library/simba/spark/lib/libsparkodbc_sb64-universal.dylib

SparkServerType=3

ServiceDiscoveryMode=No Service Discovery

Host=192.168.222.160

```
Port=10000
UID=jsmith
PWD=simba123
```

As another example, the following is an `odbc.ini` configuration file for a 32-bit connector on a Linux/AIX/Solaris machine, containing a DSN that connects to a SparkThrift Server instance and authenticates the connection using a user name and password:

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

[Sample DSN]
Driver=/opt/simba/spark/lib/32/libsparkodbc_sb32.so
SparkServerType=3
ServiceDiscoveryMode=No Service Discovery
Host=192.168.222.160
Port=10000
UID=jsmith
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 Apache Spark 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/spark/lib/libsparkodbc_sb64-universal.dylib.
```

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

```
Driver=/opt/simba/spark/lib/32/libsparkodbc_sb32.so
```

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

```
Description=Simba Apache Spark ODBC Connector
```

4. Save the `odbinst.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 `odbinst.ini` configuration file for macOS:

```
[ODBC Drivers]
```

```
Simba Apache Spark ODBC Connector=Installed
```

```
[Simba Apache Spark ODBC Connector]
```

```
Description=Simba Apache Spark ODBC Connector
```

```
Driver=/Library/simba/spark/lib/libsparkodbc_sb64-universal.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 Apache Spark ODBC Connector 32-bit=Installed

Simba Apache Spark ODBC Connector 64-bit=Installed

[Simba Apache Spark ODBC Connector 32-bit]

Description=Simba Apache Spark ODBC Connector (32-bit)

Driver=/opt/simba/spark/lib/32/libsparkodbc_sb32.so

[Simba Apache Spark ODBC Connector 64-bit]

Description=Simba Apache Spark ODBC Connector (64-bit)

Driver=/opt/simba/spark/lib/64/libsparkodbc_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](#).

For detailed information about all the connection properties that the connector supports, see [Connector Configuration Options](#).

Configuring DataStax AOSS Service Discovery on a Non-Windows Machine

You can configure the Simba Apache Spark ODBC Connector to discover Spark services through DataStax AOSS. If your AOSS endpoints require different authentication and SSL settings than the Spark service that you are connecting to, then you must also configure the connector to provide the necessary AOSS credentials and settings.

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

To configure DataStax AOSS service discovery on a non-Windows machine:

1. Set the `ServiceDiscoveryMode` property to AOSS.
2. Set the `Host` property to a comma-separated list of AOSS endpoints. Use the following format, where `[AOSS_Endpoint]` is the IP address or host name of the AOSS endpoint, and `[AOSS_Port]` is the number of the TCP port that the AOSS endpoint uses to listen for client connections:

`[AOSS_Endpoint1]:[AOSS_Port1],[AOSS_Endpoint2]:[AOSS_Port2]`

3. If the AOSS endpoints require different authentication settings than the Spark service that you are connecting to, then you must also configure the connector to provide the necessary AOSS credentials:
 - a. Set the `AOSS_AuthMech` property to 3.
 - b. Set the `AOSS_UID` property to the user name that you use to access the AOSS endpoints.
 - c. Set the `AOSS_PWD` property to the password corresponding to the user name you specified above.
4. If the AOSS endpoints require different SSL settings than the Spark service that you are connecting to, then you must also configure the connector to use the appropriate SSL settings:
 - a. Set the `AOSS_SSL` property to 1.
 - b. To allow authentication using self-signed certificates that have not been added to the list of trusted certificates, set the `AOSS_AllowSelfSignedServerCert` property to 1.
 - c. To allow the common name of a CA-issued SSL certificate to not match the host name of the AOSS endpoint, set the `AOSS_AllowHostNameCNMismatch` property to 1.
 - d. To specify the CA certificates that you want to use to verify the server, do one of the following:
 - To configure the connector to load SSL certificates from a specific `.pem` file when verifying the server, set the `AOSS_TrustedCerts` property to the full path of the `.pem` file.
 - Or, to use the trusted CA certificates `.pem` file that is installed with the connector, do not specify a value for the `AOSS_TrustedCerts` property.
 - e. To specify the minimum version of TLS to use, set the `AOSS_Min_TLS` property to the minimum version of TLS. Supported options include `1.0` for TLS 1.0, `1.1` for TLS 1.1, and `1.2` for TLS 1.2.

Configuring Authentication on a Non-Windows Machine

Some Spark Thrift Server instances are configured to require authentication for access. To connect to a Spark server, you must configure the Simba Apache Spark ODBC Connector to use the authentication mechanism that matches the access requirements of the server and provides the necessary credentials.

For information about how to determine the type of authentication your Spark server requires, see [Authentication Mechanisms](#).

You can set the connection properties for authentication in a connection string, in a DSN (in the `odbc.ini` file), or as a connector-wide setting (in the `simba.sparkodbc.ini` file). Settings in the

connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

Depending on the authentication mechanism you use, there might be additional connection attributes that you must define. For more information about the attributes involved in configuring authentication, see [Connector Configuration Options](#).

If cookie-based authentication is enabled in your Spark Server 2 database, you can specify a list of authentication cookies in the `HTTPAuthCookies` connection property. In this case, the connector authenticates the connection once based on the provided authentication credentials. It then uses the cookie generated by the server for each subsequent request in the same connection. For more information, see [HTTPAuthCookies](#).

Using No Authentication

When connecting to a Spark server of type Shark Server, you must use No Authentication. When you use No Authentication, Binary is the only Thrift transport protocol that is supported.

To configure a connection without authentication:

1. Set the `AuthMech` connection attribute to `No Authentication`.
2. If the Spark server is configured to use SSL, then configure SSL for the connection. For more information, see [Configuring SSL Verification in a Non-Windows Machine](#).

Using Kerberos

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

To configure Kerberos authentication:

1. Set the `AuthMech` connection attribute to 1.
2. Choose one:
 - To use the default realm defined in your Kerberos setup, do not set the `KrbRealm` attribute.
 - Or, if your Kerberos setup does not define a default realm or if the realm of your Spark server is not the default, then set the appropriate realm using the `KrbRealm` attribute.
3. Optionally, if you are using MIT Kerberos and a Kerberos realm is specified using the `KrbRealm` connection attribute, then choose one:
 - To have the Kerberos layer canonicalize the server's service principal name, leave the `ServicePrincipalCanonicalization` attribute set to 1.
 - Or, to prevent the Kerberos layer from canonicalizing the server's service principal name, set the `ServicePrincipalCanonicalization` attribute to 0.
4. Set the `KrbHostFQDN` attribute to the fully qualified domain name of the Spark Thrift Server host.



Note: To use the Spark server host name as the fully qualified domain name for Kerberos authentication, set `KrbHostFQDN` to `_HOST`.

5. Set the `KrbServiceName` attribute to the service name of the Spark Thrift Server.
6. To allow the connector to pass your credentials directly to the server for use in authentication, set `DelegateKrbCreds` to 1.
7. Set the `ThriftTransport` connection attribute to the transport protocol to use in the Thrift layer.

**Important:**

When using this authentication mechanism, Binary (`ThriftTransport=0`) is not supported.

8. If the Spark server is configured to use SSL, then configure SSL for the connection. For more information, see [Configuring SSL Verification in a Non-Windows Machine](#).

Using User Name

This authentication mechanism requires a user name but does not require a password. The user name labels the session, facilitating database tracking.

This authentication mechanism is available only for Spark Thrift Server on non-HDInsight distributions. Most default configurations of require User Name authentication. When you use User Name authentication, SSL is not supported and SASL is the only Thrift transport protocol available.

To configure User Name authentication:

1. Set the `AuthMech` connection attribute to 2.
2. Set the `UID` attribute to an appropriate user name for accessing the Spark server.

Using User Name And Password

This authentication mechanism requires a user name and a password.

This authentication mechanism is available only for Spark Thrift Server on non-HDInsight distributions.

To configure User Name And Password authentication:

1. Set the `AuthMech` connection attribute to 3.
2. Set the `UID` attribute to an appropriate user name for accessing the Spark server.
3. Set the `PWD` attribute to the password corresponding to the user name you provided above.
4. Set the `ThriftTransport` connection attribute to the transport protocol to use in the Thrift layer.
5. If the Spark server is configured to use SSL, then configure SSL for the connection. For more information, see [Configuring SSL Verification in a Non-Windows Machine](#).

Using Windows Azure HDInsight Emulator

This authentication mechanism is available only for Spark Thrift Server instances running in Windows Azure HDInsight Emulator. When you use this authentication mechanism, SSL is not supported and HTTP is the only Thrift transport protocol available.

To configure a connection to a Spark server in Windows Azure HDInsight Emulator:

1. Set the `AuthMech` connection attribute to 5.
2. Set the `HTTPPath` attribute to the partial URL corresponding to the Spark server.
3. Set the `UID` attribute to an appropriate user name for accessing the Spark server.
4. Set the `PWD` attribute to the password corresponding to the user name you provided above.
5. If necessary, you can create custom HTTP headers. For more information, see [http.header..](#)

Using Windows Azure HDInsight Service

This authentication mechanism is available only for Spark Thrift Server on HDInsight distributions. When you use this authentication mechanism, you must enable SSL, and HTTP is the only Thrift transport protocol available.

To configure a connection to a Spark server in Windows Azure HDInsight Service:

1. Set the `AuthMech` connection attribute to 6.
2. Set the `HTTPPath` attribute to the partial URL corresponding to the Spark server.
3. Set the `UID` attribute to an appropriate user name for accessing the Spark server.
4. Set the `PWD` attribute to the password corresponding to the user name you typed above.
5. If necessary, you can create custom HTTP headers. For more information, see [http.header..](#)
6. Configure SSL settings as needed. For more information, see [Configuring SSL Verification in a Non-Windows Machine](#).
7. Choose one:
 - To configure the connector to load SSL certificates from a specific file, set the `TrustedCerts` attribute to the path of the file.
 - Or, to use the trusted CA certificates PEM file that is installed with the connector, do not specify a value for the `TrustedCerts` attribute.

Using OAuth 2.0

Four types of authentication work flow are available when using OAuth 2.0, token pass-through, client credentials, browser based authentication, or Azure Managed Identity.

This authentication mechanism is available for Spark Thrift Server instances only. When you use OAuth 2.0 authentication, HTTP is the only Thrift transport protocol available. Client credentials and browser based authentication work flow only works when SSL is enabled.

There is a discovery mode that enables the connector to auto-fill some endpoints or configurations. The endpoint discovery is enabled by default, you can disable it by setting `EnableOIDCDiscovery=0`. You can also pass the OIDC discovery endpoint by using `OIDCDiscoveryEndpoint`. The connector automatically discovers `OAuth2AuthorizationEndPoint` and `OAuth2TokenEndPoint`.

Token Pass-through

This authentication mechanism requires a valid OAuth 2.0 access token. Be aware that access tokens typically expire after a certain amount of time, after which you must either refresh the token or obtain a new one from the server. To obtain a new access token, see [Obtaining a New Access Token](#).

To configure OAuth 2.0 token pass-through authentication:

1. Set the `AuthMech` property to 11.
2. Set the `Auth_Flow` property to 0.
3. Set the `Auth_AccessToken` property to your access token.

Obtaining a New Access Token

Once an access token expires, you can obtain a new access token for the connector.



Note: When an access token expires, the connector returns a "SQLState 08006" error.

To obtain a new access token:

1. In the connection string, set the `Auth_AccessToken` property with a new access token.
2. Call the `SQLSetConnectAttr` function with `SQL_ATTR_CREDENTIALS` (122) as the attribute and the new connection string as the value. The connector will update the current connection string with the new access token.

 **Note:** Calling the `SQLGetConnectAttr` function with `SQL_ATTR_CREDENTIALS` (122) returns the entire connection string used during connection.
3. Call the `SQLSetConnectAttr` function with `SQL_ATTR_REFRESH_CONNECTION` (123) as the attribute and `SQL_REFRESH_NOW` (-1) as the value. This signals the connector to update the access token value.
4. Retry the previous ODBC API call. After obtaining the new access token, the open connection, statements, and cursors associated with it remain valid for use.

Client Credentials

This authentication mechanism requires SSL to be enabled.

To configure OAuth 2.0 client credentials authentication:

1. Set the `AuthMech` property to 11.
2. Set the `Auth_Flow` property to 1.
3. Set the `Auth_Client_ID` to your client ID.
4. Set the `Auth_Client_Secret` to your client secret.
5. Optionally, set the `Auth_Scope` to your OAuth scope.
6. When connecting to Microsoft Synapse, set the `Auth_Tenant_ID` to your tenant id.

To configure OAuth 2.0 JWT assertion client credentials authentication:

1. Set the `AuthMech` property to 11.
2. Set the `Auth_Flow` property to 1.
3. Set the `Auth_Client_ID` to your client ID.
4. Set the `Auth_Scope` to your OAuth scope.
5. Set the `Auth_KID` to your key identifier.
6. Set the `Auth_JWT_Key_File` to the canonical path to the private key `.pem` file that matches the public key on the authentication source.
7. Optionally, set the `Auth_JWT_Key_Passphrase` to the private key's password, if it is encrypted.
8. Set `EnableOIDCDiscovery` to 1 and set `OIDCDiscoveryEndpoint` to the discovery endpoint.

Browser Based

This authentication mechanism requires SSL to be enabled.

To configure OAuth 2.0 browser based authentication:

1. Set the `AuthMech` property to 11.
2. Set the `Auth_Flow` property to 2.
3. Set the `TokenCachePassPhrase` property to a password of your choice. This is the key used for refresh token encryption.



Note: When the browser based authentication flow completes, the access token and refresh token are saved in the token cache and the connector does not need to authenticate again. For more information, see [Enable Token Cache](#).

Azure Managed Identity

This authentication mechanism requires SSL to be enabled.

To configure Azure Managed Identity authentication:

1. Set the `AuthMech` property to 11.
2. Set the `Auth_Flow` property to 3.
3. Optionally, set the `Auth_Client_ID` to user-assigned managed identity.
4. Optionally, set the `Azure_workspace_resource_id` to your assigned Resource ID.

Using an API Signing Key on a Non-Windows Machine

This authentication mechanism requires you to have credentials stored in an OCI configuration file.

The following instructions describe how to configure advanced options in a DSN and in the connector configuration tool. You can specify the connection settings described below in a DSN, in a connection string, or as connector-wide settings. Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

To configure authentication using an API signing key in Windows:

1. Set the `OCIConfigFile` property to the absolute path to the OCI configuration file to use for the connection.
2. Optionally, set the `OCIProfile` property to the name of the OCI profile to use for the connection. If no profile is specified, the connector attempts to use the profile named `DEFAULT`.
3. If the OCI profile or configuration file includes a password, set the `KeyFilePassword` property to the password.

When you initiate a connection using an API signing key, the connector uses the following behavior in case of errors:

1. If no configuration file is specified, the connector first attempts to locate the configuration file in the default location:
 - For Windows, the default location is: `%HOMEDRIVE%%HOMEPATH%\oci\config`
 - For non-Windows platforms, the default location is: `~/oci/config`
2. If the configuration file cannot be located in the default location and the `OCI_CLI_CONFIG_FILE` environment variable is specified, the connector next attempts to locate the configuration file using the value of the environment variable.
3. If the configuration file cannot be located, or the profile cannot be opened, the connector falls back to token-based authentication. For more information, see [Using Token-based Authentication on a Non-Windows Machine](#).
4. If an error occurs when using the credentials from the profile, the connector returns an error. In this case, the connector does not fall back to token-based authentication.

Using Token-based Authentication on a Non-Windows Machine

Token-based authentication is used to interactively authenticate the connection for a single session.

When you initiate a connection using token-based authentication, the connector attempts to open a web browser. You may be prompted to type your credentials in the browser.

To configure authentication using token-based authentication on a non-Windows machine:

1. Set the `OCIConfigFile` property to a path that does not contain an OCI configuration file.
2. To display a web browser used to complete the token-based authentication flow even when `SQL_DRIVER_NOPROMPT` is enabled, set the `OCIIgnoreDriverNoPrompt` property to 1.

Using MapR SASL

This authentication mechanism is only for MapR users.

To configure MapR SASL authentication:

1. Set the `AuthMech` connection attribute to 13.
2. Set the `ThriftTransport` connection attribute to the transport protocol to use in the Thrift layer.



Important: When using this authentication mechanism, Binary (`ThriftTransport=0`) is not supported.



Note: This section applies only to driver packages that includes the MapR SASL plug-ins.

Configuring SSL Verification in a Non-Windows Machine

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

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

To configure SSL verification on a non-Windows machine:

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

Configuring Server-Side Properties on a Non-Windows Machine

You can use the connector to apply configuration properties to the Spark server.

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

To configure server-side properties on a non-Windows machine:

1. To set a server-side property, use the syntax `SSP_[SSPKey]=[SSPValue]`, where `[SSPKey]` is the name of the server-side property and `[SSPValue]` is the value to specify for that property.

```
SSP_mapreduce.job.queuename=myQueue
```

i Note:

- When setting a server-side property in a connection string, it is recommended that you enclose the value in braces (`{ }`) to make sure that special characters can be properly escaped.
- For a list of all Hadoop and Spark server-side properties that your implementation supports, type `set -v` at the Spark CLI command line. You can also execute the `set -v` query after connecting using the connector.

2. To change the method that the connector uses to apply server-side properties, do one of the following:
 - To configure the connector to apply each server-side property by executing a query when opening a session to the Spark server, set the `ApplySSPWithQueries` property to 1.
 - Or, to configure the connector to use a more efficient method for applying server-side properties that does not involve additional network round-tripping, set the `ApplySSPWithQueries` property to 0.
3. To disable the connector's default behavior of converting server-side property key names to all lower-case characters, set the `LCaseSspKeyName` property to 0.

i Note:

The more efficient method is not available for Shark Server, and it might not be compatible with some Spark Thrift Server builds. If the server-side properties do not take effect when the `ApplySSPWithQueries` property is set to 0, then set it to 1.

Configuring Logging Options in a Non-Windows Machine

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



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

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

To enable logging on a non-Windows machine:

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

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

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



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

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



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

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

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

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

To disable logging on a non-Windows machine:

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

Setting Connector-Wide Configuration Options on a Non-Windows Machine

When you specify connection settings in a DSN or connection string, those settings apply only when you connect to Spark using that particular DSN or string. As an alternative, you can specify settings that apply to every connection that uses the Simba Apache Spark ODBC Connector by configuring them in the `simba.sparkodbc.ini` file.



Note: Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

To set connector-wide configuration options on a non-Windows machine:

1. In a text editor, open the `simba.sparkodbc.ini` configuration file.
2. In the `[Driver]` section, specify configuration options as key-value pairs. Start a new line for each key-value pair.

For example, to enable User Name authentication using "simba" as the user name, type the following:

```
AuthMech=2
```

```
UID=simba
```

For detailed information about all the configuration options supported by the connector, see [Connector Configuration Options](#).

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

Testing the Connection in Non-Windows Machine

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

Using the iODBC Driver Manager

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

i **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 iodbc test (or iodbc testw) 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 `iodbc test` or `iodbc testw`.
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 .

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

Using the unixODBC Driver Manager

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

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

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

To test your connection using the unixODBC driver manager:

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

- `isql [DataSourceName]`
- `iusql [DataSourceName]`

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

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

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

Authentication Mechanisms

To connect to a Spark server, you must configure the Simba Apache Spark ODBC Connector to use the authentication mechanism that matches the access requirements of the server and provides the necessary credentials. To determine the authentication settings that your Spark server requires, check the server configuration and then refer to the corresponding section below.

Shark Server

You must use No Authentication as the authentication mechanism. Shark Server instances do not support authentication.

Spark Thrift Server on an HDInsight Distribution

If you are connecting to HDInsight Emulator running in Windows Azure, then you must use the Windows Azure HDInsight Emulator mechanism.

If you are connecting to HDInsight Service running in Windows Azure, then you must use the Windows Azure HDInsight Service mechanism.

Spark Thrift Server on a non-HDInsight Distribution



Note: Most default configurations of Spark Thrift Server on non-HDInsight distributions require User Name authentication.

Configuring authentication for a connection to a Spark Thrift Server instance on a non-HDInsight Distribution involves setting the authentication mechanism, the Thrift transport protocol, and SSL support. To determine the settings that you need to use, check the following three properties in the `hive-site.xml` file in the Spark server that you are connecting to:

- `hive.server2.authentication`
- `hive.server2.transport.mode`
- `hive.server2.use.SSL`

Use the following table to determine the authentication mechanism that you need to configure, based on the `hive.server2.authentication` value in the `hive-site.xml` file:

| <code>hive.server2.authentication</code> | Authentication Mechanism |
|--|--------------------------|
| NOSASL | No Authentication |
| KERBEROS | Kerberos |
| NONE | User Name |
| LDAP | User Name and Password |

| hive.server2.authentication | Authentication Mechanism |
|-----------------------------|--------------------------|
| SAML | SAML 2.0 |

Use the following table to determine the Thrift transport protocol that you need to configure, based on the `hive.server2.authentication` and `hive.server2.transport.mode` values in the `hive-site.xml` file:

| hive.server2.authentication | hive.server2.transport.mode | Thrift Transport Protocol |
|-----------------------------|-----------------------------|---------------------------|
| NOSASL | binary | Binary |
| KERBEROS | binary or http | SASL or HTTP |
| NONE | binary or http | SASL or HTTP |
| LDAP | binary or http | SASL or HTTP |
| SAML | http | HTTP |

To determine whether SSL should be enabled or disabled for your connection, check the `hive.server2.use.SSL` value in the `hive-site.xml` file. If the value is true, then you must enable and configure SSL in your connection. If the value is false, then you must disable SSL in your connection.

For detailed instructions on how to configure authentication when using the Windows connector, see [Configuring Authentication in Windows](#).

For detailed instructions on how to configure authentication when using a non-Windows connector, see [Configuring Authentication on a Non-Windows Machine](#).

Using a Connection String

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

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

DSN Connection String Example

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

DSN=[*DataSourceName*]

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

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

DSN-less Connection String Examples

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

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

- [*AccessToken*] is your access token for authenticating the connection through the OAuth 2.0 protocol.
- [*AOSS_Endpoint*] is the IP address or host name of the DataStax AOSS endpoint.
- [*AOSS_Port*] is the number of the TCP port that the DataStax AOSS endpoint uses to listen for client connections.
- [*DomainName*] is the fully qualified domain name of the Spark server host.
- [*PortNumber*] is the number of the TCP port that the Spark server uses to listen for client connections.
- [*Server*] is the IP address or host name of the Spark server to which you are connecting.
- [*ServerURL*] is the partial URL corresponding to the Spark server.
- [*YourPassword*] is the password corresponding to your user name.
- [*YourUserName*] is the user name that you use to access the Spark server.

Connecting to a Shark Server Instance

The following is the format of a DSN-less connection string that connects to a Shark Server instance:

```
Driver=Simba Spark ODBC Driver;SparkServerType=1;  
Host=[Server];Port=[PortNumber];
```

For example:

```
Driver=Simba Spark ODBC Driver;SparkServerType=1;  
Host=192.168.222.160;Port=10000;
```

Connecting to a Standard Spark Thrift Server Instance

The following is the format of a DSN-less connection string for a standard connection to a Spark Thrift Server instance. By default, the connector is configured to connect to a Spark Thrift Server instance. Most default configurations of Spark Thrift Server require User Name authentication. When configured to provide User Name authentication, the connector uses **anonymous** as the user name by default.

```
Driver=Simba Spark ODBC Driver;Host=[Server];  
Port=[PortNumber];AuthMech=2;
```

For example:

```
Driver=Simba Spark ODBC Driver;Host=192.168.222.160;  
Port=10000;AuthMech=2;
```

Connecting to a Spark Thrift Server Instance Without Authentication

The following is the format of a DSN-less connection string that for a Spark Thrift Server instance that does not require authentication.

```
Driver=Simba Spark ODBC Driver;Host=[Server];  
Port=[PortNumber];AuthMech=0;
```

For example:

```
Driver=Simba Spark ODBC Driver;Host=192.168.222.160;  
Port=10000;AuthMech=0;
```

Connecting to a Spark Server that Requires Kerberos Authentication

The following is the format of a DSN-less connection string that connects to a Spark Thrift Server instance requiring Kerberos authentication. By default, the connector is configured to connect to a Spark Thrift Server instance.

```
Driver=Simba Spark ODBC Driver;Host=[Server];  
Port=[PortNumber];AuthMech=1;KrbRealm=[Realm];  
KrbHostFQDN=[DomainName];KrbServiceName=[ServiceName];
```

For example:

```
Driver=Simba Spark ODBC Driver;Host=192.168.222.160;  
Port=10000;AuthMech=1;KrbRealm=SIMBA;  
KrbHostFQDN=localhost.localdomain;KrbServiceName=spark;
```

Connecting to a Spark Server that Requires User Name And Password Authentication

The following is the format of a DSN-less connection string that connects to a Spark Thrift Server instance requiring User Name and Password authentication. By default, the connector is configured to connect to a Spark Thrift Server instance.

```
Driver=Simba Spark ODBC Driver;Host=[Server];  
Port=[PortNumber];AuthMech=3;UID=[YourUserName];  
PWD=[YourPassword];
```

For example:

```
Driver=Simba Spark ODBC Driver;Host=192.168.222.160;  
Port=10000;AuthMech=3;UID=simba;PWD=simba;
```

Connecting to a Spark Server in Windows Azure HDInsight Emulator

The following is the format of a DSN-less connection string that connects to a Spark Thrift Server instance running in Windows Azure HDInsight Emulator. By default, the connector is configured to connect to a Spark Thrift Server instance.

```
Driver=Simba Spark ODBC Driver;Host=[Server];  
Port=[PortNumber];AuthMech=5;UID=[YourUserName];  
PWD=[YourPassword];HTTPPath=[ServerURL];
```

For example:

```
Driver=Simba Spark ODBC Driver;Host=192.168.222.160;  
Port=10000;AuthMech=5;UID=simba;PWD=simba;  
HTTPPath=gateway/sandbox/spark;
```

Connecting to a Spark Server in Windows Azure HDInsight Service

The following is the format of a DSN-less connection string that connects to a Spark Thrift Server instance running in Windows Azure HDInsight Service. By default, the connector is configured to connect to a Spark Thrift Server instance.

```
Driver=Simba Spark ODBC Driver;Host=[Server];  
Port=[PortNumber];AuthMech=6;UID=[YourUserName];  
PWD=[YourPassword];HTTPPath=[ServerURL];
```

For example:

```
Driver=Simba Spark ODBC Driver;Host=192.168.222.160;  
Port=10000;AuthMech=6;UID=simba;PWD=simba;  
HTTPPath=gateway/sandbox/spark;
```

Connecting to a Spark Server that Requires OAuth 2.0 Authentication

The following is the format of a DSN-less connection string that connects to a Spark Thrift Server instance requiring OAuth 2.0 authentication. By default, the connector is configured to connect to a

Spark Thrift Server instance. Browser based authentication workflow only works when SSL is enabled.

Token pass-through

```
Driver=Simba Spark ODBC Driver;Host=[/Server];  
Port=[/PortNumber];AuthMech=11;Auth_Flow=0;Auth_AccessToken=[AccessToken];ThriftTransport=2;
```

For example, using token pass-through authentication:

```
Driver=Simba Spark ODBC Driver;Host=192.168.222.160;  
Port=10000;AuthMech=11;Auth_Flow=0;Auth_  
AccessToken=P9QcyQ7prK2LwUMZMpFQ4R+6jd;ThriftTransport=2;
```

Browser based

```
Driver=SimbaSparkODBC Driver;Host=[/Server];  
Port=[/PortNumber];AuthMech=11;Auth_Flow=2;ThriftTransport=2;SSL=1;
```

For example, using browser based authentication:

```
Driver=SimbaSpark ODBC Driver;Host=192.168.222.160;  
Port=10000;AuthMech=11;Auth_Flow=2;ThriftTransport=2;SSL=1;
```

Connecting using Dynamic Service Discovery

The following is the format of a DSN-less connection string that discovers Spark services via the DataStax AOSS service.

```
Driver=Simba Spark ODBC Driver;ServiceDiscoveryMode=AOSS;  
Host=[AOSS_Endpoint1]:[AOSS_Port1], [AOSS_Endpoint2]:[AOSS_Port2];
```

For example:

```
Driver=Simba Spark ODBC Driver;ServiceDiscoveryMode=AOSS;  
Host=192.168.222.160:9077, 192.168.222.165:9077;
```

Connecting to a DFI Server Using an API Signing Key

The following is the format of a DSN-less connection string that connects to a DFI Server instance with an API signing key:

```
Driver=Simba Spark ODBC Driver;Host=[/Server];OCIConfigFile=[ConfigFile];
```

For example:

```
Driver=Simba Spark ODBC Driver;Host=192.168.222.160;OCIConfigFile="C:\config.cnf";
```

Connecting to a DFI Server Using a Token

The following is the format of a DSN-less connection string that connects to a DFI server using token-based authentication:

```
Driver=Simba Spark ODBC Driver;Host=[/Server];OCIConfigFile="";
```

For example:

```
Driver=Simba Spark ODBC Driver;Host=192.168.222.160;OCIConfigFile="";
```

Features

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

- [SQL Connector for HiveQL](#)
- [Data Types](#)
- [Timestamp Function Support](#)
- [Catalog and Schema Support](#)
- [spark_system Table](#)
- [Server-Side Properties](#)
- [Get Tables With Query](#)
- [Active Directory](#)
- [Write-back](#)
- [Dynamic Service Discovery using DataStax AOSS](#)
- [Security and Authentication](#)

SQL Connector for HiveQL

The native query language supported by Spark is HiveQL. For simple queries, HiveQL is a subset of SQL-92. However, the syntax is different enough that most applications do not work with native HiveQL.

To bridge the difference between SQL and HiveQL, the SQL Connector feature translates standard SQL-92 queries into equivalent HiveQL queries. The SQL Connector performs syntactical translations and structural transformations. For example:

- **Quoted Identifiers:** The double quotes (") that SQL uses to quote identifiers are translated into back quotes (`) to match HiveQL syntax. The SQL Connector needs to handle this translation because even when a connector reports the back quote as the quote character, some applications still generate double-quoted identifiers.
- **Table Aliases:** Support is provided for the AS keyword between a table reference and its alias, which HiveQL normally does not support.
- **JOIN, INNER JOIN, and CROSS JOIN:** SQL JOIN, INNER JOIN, and CROSS JOIN syntax is translated to HiveQL JOIN syntax.
- **TOP N/LIMIT:** SQL TOP N queries are transformed to HiveQL LIMIT queries.

Data Types

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

The following table lists the supported data type mappings.

| Spark Type | SQL Type |
|---------------|--------------------|
| BIGINT | SQL_BIGINT |
| BINARY | SQL_VARBINARY |
| BOOLEAN | SQL_BIT |
| CHAR(n) | SQL_CHAR |
| DATE | SQL_TYPE_DATE |
| DECIMAL | SQL_DECIMAL |
| DECIMAL(p,s) | SQL_DECIMAL |
| DOUBLE | SQL_DOUBLE |
| FLOAT | SQL_REAL |
| INT | SQL_INTEGER |
| SMALLINT | SQL_SMALLINT |
| STRING | SQL_VARCHAR |
| TIMESTAMP | SQL_TYPE_TIMESTAMP |
| TIMESTAMP_NTZ | SQL_TYPE_TIMESTAMP |
| TINYINT | SQL_TINYINT |
| VARCHAR(n) | SQL_VARCHAR |



Note: The aggregate types (ARRAY, MAP, and STRUCT) are not supported. Columns of aggregate types are treated as STRING columns.

Timestamp Function Support

The Simba Apache Spark ODBC Connector supports the following ODBC functions for working with data of type TIMESTAMP:

- **TIMESTAMPADD:** You can call this function to increment a TIMESTAMP value by a specified interval of time.
- **TIMESTAMPDIFF:** You can call this function to calculate the interval of time between two specified TIMESTAMP values.

The types of time intervals that are supported for these functions might vary depending on the Spark server version that you are connecting to. To return a list of the intervals supported for TIMESTAMPADD, call the SQLGetInfo catalog function using SQL_TIMEDATE_ADD_INTERVALS as the argument. Similarly, to return a list of the intervals supported for TIMESTAMPDIFF, call SQLGetInfo using SQL_TIMEDATE_DIFF_INTERVALS as the argument.



Note: The SQL_TSI_FRAC_SECOND interval is not supported by Spark.

Catalog and Schema Support

The Simba Apache Spark ODBC Connector supports both catalogs and schemas to make it easy for the connector to work with various ODBC applications. For servers that do not support multiple catalogs, the connector provides a synthetic catalog named SPARK under which all of the schemas/databases are organized.



Note: When connecting to a server that supports multiple catalogs, and the server is not an IDL server, the connector no longer reports the catalog for schemas and tables as SPARK. The Spark server now reports the catalog. The only exception is the spark_system table which remains in the SPARK catalog. The multiple catalog feature is enabled when connecting to an IDL cluster.

spark_system Table

A pseudo-table called spark_system can be used to query for Spark cluster system environment information. The pseudo-table is under the pseudo-schema called spark_system. The table has two STRING type columns, envkey and envvalue. Standard SQL can be executed against the spark_system table. For example:

```
SELECT * FROM SPARK.spark_system.spark_system WHERE envkey LIKE '%spark%'
```

The above query returns all of the Spark system environment entries whose key contains the word "spark". A special query, `set -v`, is executed to fetch system environment information. Some versions of Spark do not support this query. For versions of Spark that do not support querying system environment information, the connector returns an empty result set.

Server-Side Properties

The Simba Apache Spark ODBC Connector allows you to set server-side properties via a DSN. Server-side properties specified in a DSN affect only the connection that is established using the DSN.

You can also specify server-side properties for connections that do not use a DSN. To do this, use the Simba Apache Spark ODBC Connector Configuration tool that is installed with the Windows version of the connector, or set the appropriate configuration options in your connection string or the `simba.sparkodbc.ini` file. Properties specified in the connector configuration tool or the `simba.sparkodbc.ini` file apply to all connections that use the Simba Apache Spark ODBC Connector.

For more information about setting server-side properties when using the Windows connector, see [Configuring Server-Side Properties in Windows](#). For information about setting server-side properties when using the connector on a non-Windows platform, see [Configuring Server-Side Properties on a Non-Windows Machine](#).

Get Tables With Query

The Get Tables With Query configuration option allows you to choose whether to use the SHOW TABLES query or the GetTables API call to retrieve table names from a database.

Active Directory

The Simba Apache Spark ODBC Connector supports Active Directory Kerberos in Windows. There are two prerequisites for using Active Directory Kerberos in Windows:

- MIT Kerberos is not installed on the client Windows machine.
- The MIT Kerberos Hadoop realm has been configured to trust the Active Directory realm so that users in the Active Directory realm can access services in the MIT Kerberos Hadoop realm.

Write-back

The Simba Apache Spark ODBC Connector supports translation for the following syntax when connecting to a Spark Thrift Server instance that is running Spark 1.3 or later:

- INSERT
- CREATE
- DROP

Spark does not support UPDATE or DELETE syntax.

If the statement contains non-standard SQL-92 syntax, then the connector is unable to translate the statement to SQL and instead falls back to using HiveQL.

Dynamic Service Discovery using DataStax AOSS

The Simba Apache Spark ODBC Connector can be configured to discover services via the DataStax AOSS service.

For information about configuring this feature in the Windows connector, see the following:

- [Creating a Data Source Name in Windows](#)
- [Configuring a DSN-less Connection in Windows](#)
- [Configuring AOSS Options in Windows](#)

For information about configuring this feature in a non-Windows connector, see [Configuring DataStax AOSS Service Discovery on a Non-Windows Machine](#).

Security and Authentication

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



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

The connector provides mechanisms that enable you to authenticate your connection using the Kerberos protocol, the OAuth 2.0 protocol, an API signing key, token-based authentication, your Spark user name only, or your Spark user name and password. You can also authenticate a connection to an HDInsight distribution in Windows Azure. You must use the authentication mechanism that matches the security requirements of the Spark server. For information about determining the appropriate authentication mechanism to use based on the Spark server configuration, see [Authentication Mechanisms](#). For detailed connector configuration instructions, see [Configuring Authentication in Windows](#) or [Configuring Authentication on a Non-Windows Machine](#).

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

- No identity verification
- One-way authentication
- Two-way authentication

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

Connector Configuration Options

Connector Configuration Options lists the configuration options available in the Simba Apache Spark 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 are available in the Simba Apache Spark ODBC Connector Configuration tool and the following dialog boxes:

- Simba Spark ODBC Connector DSN Setup
- AOSS Options
- OAuth Options
- Advanced Options
- HTTP Proxy Options
- Server Side Properties
- SSL Options
- HTTP Properties

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

**Note:**

If you are using the connector on a non-Windows machine, you can set connector configuration properties in a connection string, in a DSN (in the `odbc.ini` file), or as a connector-wide setting (in the `simba.sparkodbc.ini` file). Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

Configuration Options Appearing in the User Interface

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

- [Accept Undetermined Revocation](#)
- [Access Token](#)
- [Log Level](#)
- [Log Path](#)

- Allow Common Name Host Name Mismatch
- Allow Common Name Host Name Mismatch (for AOSS)
- Allow Self-Signed Server Certificate
- Allow Self-Signed Server Certificate (for AOSS)
- Apply Properties With Queries
- Async Exec Poll Interval
- Authentication Flow
- Auth_RefreshToken
- Binary Column Length
- Canonicalize Principal FQDN
- CheckCertificate Revocation
- Check Certificate Revocation (for AOSS)
- Client Certificate File
- Client ID
- Client Private Key File
- Client Private Key Password
- Client Secret
- Convert Key Name to Lower Case
- Database
- Decimal Column Scale
- Default String Column Length
- Delegate Kerberos Credentials
- Delegation UID
- Max Bytes Per Fetch Request
- Max File Size
- Max Number Files
- Mechanism
- Minimum TLS Version
- Minimum TLS Version (for AOSS)
- OAuth Scope
- OCI Config File
- OCI Profile
- Password
- Password (for AOSS)
- Port
- ProxyAuth
- Proxy Host
- ProxyKrbHostFQDN
- ProxyKrbREALM
- ProxyKrbServiceName
- Proxy Ignore List
- Proxy Password
- Proxy Port
- Proxy Username
- Realm
- Rows Fetched Per Block
- Save Password (Encrypted)
- Service Discovery Mode

- Driver Config Take Precedence
- Enable Auto Reconnect
- Enable SSL
- Enable SSL (for AOSS)
- Enable Token Cache
- Enable Translation For CTAS
- Fast SQLPrepare
- Get Tables With Query
- Host(s)
- Host FQDN
- HTTP Path
- Ignore SQL_DRIVER_NOPROMPT
- Ignore SQL_DRIVER_NOPROMPT (OAuth2)
- Invalid Session Auto Recover
- JWT Key Identifier
- JWT Private Key Password
- JWT Private Key Path
- Key File Password
- Service Name
- Show System Table
- Socket Timeout
- Spark Server Type
- Status Endpoint Authentication
- Thrift Transport
- TokenCachePassPhrase
- TokenRenewLimit
- Trusted Certificates
- Trusted Certificates (for AOSS)
- Two-Way SSL
- Unicode SQL Character Types
- Use Async Exec
- Use JWT Assertion
- Use Native Query
- Use OIDC Discovery Endpoint
- Use Only SSPI
- Use Proxy Server
- Use System Trust Store
- Use System Trust Store (for AOSS)
- User Name
- User Name (for AOSS)

Accept Undetermined Revocation

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

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

| Key Name | Default Value | Required |
|------------------------------|---------------|----------|
| AcceptUndeterminedRevocation | 0 | No |

Access Token

The access token for authenticating the connection through the OAuth 2.0 protocol.



Note: In case of unixODBC, when the Auth_AccessToken line length is longer than the maximum limit of 1000, add the following in your odbc.ini file:

In the ODBC section:

- Auth_AccessToken=(your access token)

In the DSN section:

- ConfigsFromFileDSN=Auth_AccessToken
- FILEDSNPATH=(Full path of the odbc.ini file)
- Auth_AccessToken=(your access token)

If you have multiple DSN configured in your odbc.ini file and each of them require a different Auth_AccessToken, you can add the Auth_AccessToken to the ODBC section of a different ini file, and configure the FILEDSNPATH in your DSN to point to this ini file.

| Key Name | Default Value | Required |
|------------------|---------------|--|
| Auth_AccessToken | None | Yes, if the authentication mechanism is OAuth 2.0 (11) and the work flow is Token Passthrough (0). |

Allow Common Name Host Name Mismatch

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

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

This setting is applicable only when SSL is enabled.

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

| Key Name | Default Value | Required |
|--------------------------------------|---------------|----------|
| <code>AllowHostNameCNMismatch</code> | Clear (0) | No |

Allow Common Name Host Name Mismatch (for AOSS)

This property specifies whether the connector requires the name of the CA-issued SSL certificate to match the host name of the AOSS endpoint.

- Enabled (1): The connector allows the names to mismatch.
- Disabled (0): The connector requires the names to match.

i **Note:** This property is applicable only when AOSS dynamic service discovery is enabled, and SSL is enabled for AOSS connections.

| Key Name | Default Value | Required |
|---|--|----------|
| <code>AOSS_AllowHostNameCNMismatch</code> | The value being used for the non-AOSS version of this property, which defaults to Clear (0). | No |

Allow Self-Signed Server Certificate

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

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

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

| Key Name | Default Value | Required |
|---------------------------|---------------|----------|
| AllowSelfSignedServerCert | Clear (0) | No |

Allow Self-Signed Server Certificate (for AOSS)

| Key Name | Default Value | Required |
|--------------------------------|--|----------|
| AOSS_AllowSelfSignedServerCert | The value being used for the non-AOSS version of this property, which defaults to Clear (0). | No |

Description

This property specifies whether the connector allows the AOSS endpoint to use self-signed SSL certificates.

- Enabled (1): The connector allows self-signed certificates.



Important: When this property is enabled, SSL verification is disabled. The connector does not verify the endpoint's certificate against the trust store, and does not verify if the endpoint's host name matches the common name or subject alternative names in the endpoint's certificate.

- Disabled (0): The connector does not allow self-signed certificates.



Note: This property is applicable only when AOSS dynamic service discovery is enabled, and SSL is enabled for AOSS connections.

Apply Properties With Queries

This option specifies how the connector applies server-side properties.

- Enabled (1): The connector applies each server-side property by executing a `set SSPKey=SSPValue` query when opening a session to the Spark server.
- Disabled (0): The connector uses a more efficient method for applying server-side properties that does not involve additional network round-tripping. However, some Spark Thrift Server builds are not compatible with the more efficient method.



Note: When connecting to a Shark Server instance, this option is always enabled.

| Key Name | Default Value | Required |
|---------------------|---------------|----------|
| ApplySSPWithQueries | Selected (1) | No |

Async Exec Poll Interval

The time in milliseconds between each poll for the query execution status.

"Asynchronous execution" refers to the fact that the RPC call used to execute a query against Spark is asynchronous. It does not mean that ODBC asynchronous operations are supported.



Note: This option is applicable only to HDInsight clusters.

| Key Name | Default Value | Required |
|-----------------------|---------------|----------|
| AsyncExecPollInterval | 100 | No |

Authentication Flow

This option specifies the type of OAuth authentication flow that the connector uses when the Mechanism option is set to OAuth 2.0 (or when `AuthMech` is set to 11).

When this option is set to Token Passthrough (0), the connector uses the access token specified by the Access Token (`Auth_AccessToken`) option to authenticate the connection to the server. For more information, see [Access Token](#).

When this option is set to Client Credentials (1), the connector uses the client credentials to authenticate the connection to the server.

When this option is set to Browser Based Authorization Code (2), the connector uses the browser based authorization code flow to authenticate the connection to the server.

| Key Name | Default Value | Required |
|-----------|-----------------------|----------|
| Auth_Flow | Token Passthrough (0) | No |

Auth_RefreshToken

The `Auth_RefreshToken` property enables the driver to automatically refresh the authentication token using the specified refresh token.

| Key Name | Default Value | Required |
|-------------------|---------------|----------|
| Auth_RefreshToken | Empty | No |

Binary Column Length

The maximum data length for BINARY columns.

By default, the columns metadata for Spark does not specify a maximum data length for BINARY columns.

| Key Name | Default Value | Required |
|--------------------|---------------|----------|
| BinaryColumnLength | 32767 | No |

Canonicalize Principal FQDN

This option specifies whether the Kerberos layer canonicalizes the host FQDN in the server's service principal name.

- Enabled (1): The Kerberos layer canonicalizes the host FQDN in the server's service principal name.
- Disabled (0): The Kerberos layer does not canonicalize the host FQDN in the server's service principal name.



Note: This option only affects MIT Kerberos, and is ignored when using Active Directory Kerberos. It can only be disabled if the Kerberos Realm or `KrbRealm` key is specified.

| Key Name | Default Value | Required |
|----------------------------------|---------------|----------|
| ServicePrincipalCanonicalization | Selected (1) | No |

CheckCertificate Revocation

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

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

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



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

| Key Name | Default Value | Required |
|---------------------|---------------|----------|
| CheckCertRevocation | Selected (1) | No |

Check Certificate Revocation (for AOSS)

This property specifies whether, when connecting to an AOSS endpoint with SSL enabled, the connector checks to see if a certificate has been revoked while retrieving a certificate chain from the Windows Trust Store.

- Enabled (1): The connector checks for certificate revocation.
- Disabled (0): The connector does not check for certificate revocation.


Note:

This property is applicable only when all of the following is true:

- AOSS dynamic service discovery is enabled.
- SSL is enabled for AOSS connections.
- The connector is configured to use a CA certificate from the Windows Trust Store (see [Use System Trust Store \(for AOSS\)](#)).

| Key Name | Default Value | Required |
|--------------------------|---|----------|
| AOSS_CheckCertRevocation | The value being used for the non-AOSS version of this property, which defaults to Selected (1). | No |

Client Certificate File

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



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

| Key Name | Default Value | Required |
|------------|---------------|----------|
| ClientCert | None | No |

Client ID

The client ID used in OAuth connections.

| Key Name | Default Value | Required |
|----------------|---------------|----------|
| Auth_Client_ID | None | No |

Client Private Key File

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

If the private key file is protected with a password, then provide the password using the connector configuration option [Client Private Key Password](#).



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

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

Client Private Key Password

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

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

Client Secret

The client secret associated with the client ID used in OAuth connections.

| Key Name | Default Value | Required |
|--------------------|---------------|----------|
| Auth_Client_Secret | None | No |

Convert Key Name to Lower Case

This option specifies whether the connector converts server-side property key names to all lower-case characters.

- Enabled (1): The connector converts server-side property key names to all lower-case characters.
- Disabled (0): The connector does not modify the server-side property key names.

| Key Name | Default Value | Required |
|-----------------|---------------|----------|
| LCaseSspKeyName | Selected (1) | No |

Database

The name of the database schema to use when a schema is not explicitly specified in a query. You can still issue queries on other schemas by explicitly specifying the schema in the query.

 **Note:** To inspect your databases and determine the appropriate schema to use, at the Spark command prompt, type `show databases`.

| Key Name | Default Value | Required |
|----------|---------------|----------|
| Schema | default | No |

Decimal Column Scale

The maximum number of digits to the right of the decimal point for numeric data types.

| Key Name | Default Value | Required |
|--------------------|---------------|----------|
| DecimalColumnScale | 10 | No |

Default String Column Length

The maximum number of characters that can be contained in STRING columns.

By default, the columns metadata for Spark does not specify a maximum length for STRING columns.

| Key Name | Default Value | Required |
|---------------------------|---------------|----------|
| DefaultStringColumnLength | 255 | No |

Delegate Kerberos Credentials

This option specifies whether your Kerberos credentials are forwarded to the server and used for authentication.



Note: This option is only applicable when Authentication Mechanism is set to Kerberos (AuthMech=1).

| Key Name | Default Value | Required |
|------------------|---------------|----------|
| DelegateKrbCreds | Clear (0) | No |

Delegation UID

If a value is specified for this setting, the connector delegates all operations against Spark to the specified user, rather than to the authenticated user for the connection.



Note: This option is applicable only when connecting to a Spark Thrift Server instance that supports this feature.

| Key Name | Default Value | Required |
|---------------|---------------|----------|
| DelegationUID | None | No |

Driver Config Take Precedence

This option specifies whether connector-wide configuration settings take precedence over connection and DSN settings.

- Enabled (1): Connector-wide configurations take precedence over connection and DSN settings.
- Disabled (0): Connection and DSN settings take precedence instead.

| Key Name | Default Value | Required |
|----------------------------|---------------|----------|
| DriverConfigTakePrecedence | Clear (0) | No |

Enable Auto Reconnect

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

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

| Key Name | Default Value | Required |
|---------------|---------------|----------|
| AutoReconnect | Selected (1) | Yes |

Enable SSL

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

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

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



Note:

- This option is applicable only when connecting to a Sparkserver that supports SSL.
- If you selected User Name or Windows Azure HDInsight Emulator as the authentication mechanism, SSL is not available.

| Key Name | Default Value | Required |
|----------|---------------|----------|
| SSL | Clear (0) | No |

Enable SSL (for AOSS)

This property specifies whether the connector communicates with AOSS endpoints through SSL-enabled sockets.

- Enabled (1): The connector connects to SSL-enabled sockets.
- Disabled (0): The connector does not connect to SSL-enabled sockets.

 **Note:** This property is applicable only when AOSS dynamic service discovery is enabled.

| Key Name | Default Value | Required |
|----------|--|----------|
| AOSS_SSL | The value being used for the non-AOSS version of this property, which defaults to Clear (0). | No |

Enable Token Cache

This option specifies whether the connector enables or disables token cache for OAuth2 Browser Based authentication.

- Enabled (1): The connector enables the token cache for OAuth2 Browser Based authentication.
- Disabled (0): The connector disables the token cache for OAuth2 Browser Based authentication.

| Key Name | Default Value | Required |
|------------------|---------------|----------|
| EnableTokenCache | Enabled (1) | No |

Enable Translation For CTAS

This property specifies whether the connector performs a query translation for the CREATE TABLE AS SELECT (CTAS) syntax.

- Enabled (1): The connector performs a query translation for the CTAS syntax.
- Disabled (0): The connector does not perform a query translation for the CTAS syntax.

| Key Name | Default Value | Required |
|--------------------------|---------------|----------|
| EnableTranslationForCTAS | Enabled (1) | No |

Fast SQLPrepare

This option specifies whether the connector defers query execution to SQLExecute.

- Enabled (1): The connector defers query execution to SQLExecute.
- Disabled (0): The connector does not defer query execution to SQLExecute.



Note: When using Native Query mode, the connector executes the HiveQL query to retrieve the result set metadata for SQLPrepare. As a result, SQLPrepare might be slow. If the result set metadata is not required after calling SQLPrepare, then enable Fast SQLPrepare.

| Key Name | Default Value | Required |
|----------------|---------------|----------|
| FastSQLPrepare | Clear (0) | No |

Get Tables With Query

This option specifies whether the connector uses the SHOW TABLES query or the GetTables Thrift API call to retrieve table names from the database.

- Enabled (1): The connector uses the SHOW TABLES query to retrieve table names.
- Disabled (0): The connector uses the GetTables Thrift API call to retrieve table names.



Note:

- This option is applicable only when connecting to a Spark Thrift Server instance.
- On Spark Server 3.0 and earlier, table names are always retrieved using the SHOW TABLES query because the GetTables API call is not supported on earlier versions.

| Key Name | Default Value | Required |
|--------------------|---------------|----------|
| GetTablesWithQuery | 0 | No |

Host(s)

The IP address or host name of the Spark server.

Alternatively, if Service Discovery Mode is enabled, specify a comma-separated list of AOSS endpoints. Use the following format, where *[AOSS_Endpoint]* is the IP address or host name of the AOSS endpoint, and *[AOSS_Port]* is the number of the TCP port that the AOSS endpoint uses to listen for client connections:

[AOSS_Endpoint1]:[AOSS_Port1],[AOSS_Endpoint2]:[AOSS_Port2]

| Key Name | Default Value | Required |
|----------|---------------|----------|
| Host | None | Yes |

Host FQDN

The fully qualified domain name of the Spark Thrift Server host.

When the value of Host FQDN is `_HOST`, the connector uses the Spark server host name as the fully qualified domain name for Kerberos authentication.

| Key Name | Default Value | Required |
|-------------|--------------------|----------|
| KrbHostFQDN | <code>_HOST</code> | No |

HTTP Path

The partial URL corresponding to the Spark server.

The connector forms the HTTP address to connect to by appending the HTTP Path value to the host and port specified in the DSN or connection string. For example, to connect to the HTTP address `http://localhost:10002/gateway/sandbox/spark/version`, you would set HTTP Path to `/gateway/sandbox/spark/version`.

| Key Name | Default Value | Required |
|----------|--|----------|
| HTTPPath | <p><code>/spark</code> if using Windows Azure HDInsight Service (6).</p> <p><code>/</code> if using non-Windows Azure HDInsight Service with Thrift Transport set to HTTP (2).</p> | No |

Ignore SQL_DRIVER_NOPROMPT

| Key Name | Default Value | Required |
|-------------------------|---------------|----------|
| OCIIgnoreDriverNoPrompt | Clear (0) | No |

Description

This property specifies whether the connector displays a web browser for token-based authentication when DFI is selected as the Spark Server Type and when the application makes a SQLDriverConnect API call with a SQL_DRIVER_NOPROMPT flag to the connector.

- Enabled (1): The connector displays the web browser used to complete the token-based authentication flow even when SQL_DRIVER_NOPROMPT is enabled.
- Disabled (0): The connector does not display a web browser when SQL_DRIVER_NOPROMPT is enabled.

Ignore SQL_DRIVER_NOPROMPT (OAuth2)

| Key Name | Default Value | Required |
|---------------------------|---------------|----------|
| OAuthIgnoreDriverNoPrompt | Selected (1) | No |

Description

When using browser based OAuth2 authentication, this option specifies whether the connector displays a web browser when the application makes a SQLDriverConnect API call with a SQL_DRIVER_NOPROMPT flag to the connector.

- Enabled (1): The connector displays the web browser used to complete the browser based authentication flow even when SQL_DRIVER_NOPROMPT is passed.
- Disabled (0): The connector does not display a web browser.

Invalid Session Auto Recover

This option specifies whether the connector automatically opens a new session when the existing session is no longer valid.

- Enabled (1): The connector automatically opens a new session when the existing session is no longer valid.
- Disabled (0): The connector does not automatically open new sessions.

 **Note:** This option is applicable only when connecting to Spark Thrift Server.

| Key Name | Default Value | Required |
|---------------------------|---------------|----------|
| InvalidSessionAutoRecover | Selected (1) | No |

JWT Key Identifier

The key identifier used in JWT client credential connections.

 **Note:** This option is applicable only when Use JWT Assertion is enabled.

| Key Name | Default Value | Required |
|----------|---------------|----------|
| Auth_KID | None | No |

JWT Private Key Password

The password for the encrypted private key file used in JWT client credential connections.

 **Note:** This option is applicable only when Use JWT Assertion is enabled.

| Key Name | Default Value | Required |
|---------------------|---------------|----------|
| Auth_JWT_Passphrase | None | No |

JWT Private Key Path

The canonical path to the `.pem` private key file used in JWT client credential connections.



Note:

This option is applicable only when `Use JWT Assertion` is enabled.

This private key must match the public key on the authentication source. If the private key is encrypted, a value for `Auth_JWT_Passphrase` must be provided.

| Key Name | Default Value | Required |
|-------------------|---------------|----------|
| Auth_JWT_Key_File | None | No |

Key File Password

The password for the key file if the key file is password protected.

| Key Name | Default Value | Required |
|-----------------|---------------|----------|
| keyFilePassword | None | 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 `simbasparkodbcdriver.log` file that logs connector activity that is not specific to a connection.
- A `simbasparkodbcdriver_connection_[Number].log` file for each connection made to the database, where `[Number]` is a number that identifies each log file. This file logs connector activity that is specific to the connection.

| Key Name | Default Value | Required |
|-----------------------|---------------|----------|
| <code>LogLevel</code> | 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 |
|----------------------|---------------|-----------------------------|
| <code>LogPath</code> | None | Yes, if logging is enabled. |

Max Bytes Per Fetch Request

When connecting to a server that supports serializing the result set data in Arrow format, this property specifies the maximum number of bytes to retrieve from the server for every `FetchResults` API call.

**Note:**

This option is applicable only when connecting to a server that supports result set data serialized in arrow format.

The value must be specified in one of the following:

- B (bytes)
- KB (kilobytes)
- MB (megabytes)
- GB (gigabytes)

By default, the file size is in B (bytes).

When the result set type is ARROW_BASED_SET, the server will cap the rowset size at 10 MB even when the connector indicates that it can consume more than 10 MB of result set data for each FetchResults API call.

| Key Name | Default Value | Required |
|-------------------------|---------------|----------|
| MaxBytesPerFetchRequest | 300 MB | No |

Max File Size

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

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



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

| Key Name | Default Value | Required |
|-------------|---------------|----------|
| LogFileSize | 20971520 | No |

Max Number Files

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



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

| Key Name | Default Value | Required |
|--------------|---------------|----------|
| LogFileCount | 50 | No |

Mechanism

The authentication mechanism to use.

Select one of the following settings, or set the key to the corresponding number:

- No Authentication (0)
- Kerberos (1)
- User Name (2)
- User Name And Password (3)
- JWT (JWT or 8)
- Windows Azure HDInsight Emulator (5)
- Windows Azure HDInsight Service (6)
- OAuth 2.0 (11)
- MapR SASL (13)

| Key Name | Default Value | Required |
|----------|--|----------|
| AuthMech | No Authentication (0 if you are connecting to Spark Server 1. User Name (2) if you are connecting to Spark Server 2. | No |

Minimum TLS Version

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

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

| Key Name | Default Value | Required |
|----------|---------------|----------|
| Min_TLS | TLS 1.2 (1.2) | No |

Minimum TLS Version (for AOSS)

| Key Name | Default Value | Required |
|--------------|--|----------|
| AOSS_Min_TLS | The value being used for the non-AOSS version of this property, which defaults to TLS 1.2 (1.2). | No |

Description

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

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



Note: This property is applicable only when AOSS dynamic service discovery is enabled, and SSL is enabled for AOSS connections.

OAuth Scope

The scope used in the OAuth 2.0 client credentials connection.

| Key Name | Default Value | Required |
|------------|---------------|----------|
| Auth_Scope | None | No |

OCI Config File

The absolute path to the OCI configuration file to use for the connection.



Note: If this property is specified, the connector ignores the OCI_CLI_CONFIG_FILE environment variable when attempting to locate the configuration file.

| Key Name | Default Value | Required |
|---------------|---------------|----------|
| OCIConfigFile | None | No |

OCI Profile

The name of the OCI profile to use for the connection. The connector retrieves the named profile from the configuration file, and uses its credentials for the connection.

If the named profile cannot be opened, the connector switches to token-based authentication.

| Key Name | Default Value | Required |
|------------|---------------|----------|
| OCIProfile | DEFAULT | No |

OIDC Discovery Endpoint

The OIDC discovery endpoint.

| |
|--|
|  Note: This option is applicable only when Use OIDC Discovery Endpoint is enabled. |
|--|

| Key Name | Default Value | Required |
|-----------------------|---------------|----------|
| OIDCDiscoveryEndpoint | None | No |

Password

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

| Key Name | Default Value | Required |
|----------|---------------|---|
| PWD | None | Yes, if the authentication mechanism is User Name And Password (3), Windows Azure HDInsight Emulator (5), or Windows Azure HDInsight Service (6). |

Password (for AOSS)

The password corresponding to the user name that you provided using the property [User Name \(for AOSS\)](#).

| |
|---|
|  Note: This property is applicable only when AOSS dynamic service discovery is enabled. |
|---|

| Key Name | Default Value | Required |
|----------|--|----------|
| AOSS_PWD | The value being used for the non-AOSS version of this property, which does not have a default value. | No |

Port

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

| Key Name | Default Value | Required |
|----------|--|---|
| Port | <ul style="list-style-type: none"> ■ non-HDInsight clusters: 10000 ■ Windows Azure HDInsight Emulator: 10001 ■ Windows Azure HDInsight Service: 443 | Yes, if Service Discovery Mode is disabled. |

ProxyAuth

This option specifies whether the proxy server that you are connecting to requires authentication.

- 0: The proxy server that you are connecting to does not require authentication.
- 1: The proxy server that you are connecting to requires basic authentication.
- 2: The proxy server that you are connecting to requires proxy Kerberos authentication.

| Key Name | Default Value | Required |
|---------------|---------------|----------|
| ProxyAuthType | 0 | 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. |

ProxyKrbHostFQDN

The connector uses the host FQDN of the ProxyHost.

| Key Name | Default Value | Required |
|------------------|---------------|----------|
| ProxyKrbHostFQDN | None | No |

ProxyKrbServiceName

This option specifies the service name for kerberos proxy.

| Key Name | Default Value | Required |
|---------------------|---------------|----------|
| ProxyKrbServiceName | HTTP | No |

ProxyKrbREALM

The connector takes the default realm of the system.

| Key Name | Default Value | Required |
|---------------|---------------|----------|
| ProxyKrbREALM | None | No |

Proxy Ignore List

This option specifies whether the connector allows the hosts or domains that do not use proxy. For example: `ProxyIgnoreList=localhost,127.0.0.1`

| Key Name | Default Value | Required |
|-----------------|---------------|----------|
| ProxyIgnoreList | Empty string | No |

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 Password

The password that you use to access the proxy server.

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

Proxy Username

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

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

Realm

The realm of the Spark Thrift Server host.

If your Kerberos configuration already defines the realm of the Spark Thrift Server host as the default realm, then you do not need to configure this option.

| Key Name | Default Value | Required |
|----------|---|----------|
| KrbRealm | Depends on your Kerberos configuration. | No |

Rows Fetched Per Block

The maximum number of rows that a query returns at a time.

Valid values for this setting include any positive 32-bit integer. However, testing has shown that performance gains are marginal beyond the default value of 10000 rows.

| Key Name | Default Value | Required |
|---------------------|---------------|----------|
| RowsFetchedPerBlock | 10000 | No |

Save Password (Encrypted)

This option specifies whether the password is saved in the registry.

- Enabled: The password is saved in the registry.
- Disabled: The password is not saved in the registry.



Note: This option is available only on the Windows connector. It appears in the SimbaSpark ODBC Connector DSN Setup dialog box and the SSL Options dialog box. The password is obscured (not saved in plain text). However, it is still possible for the encrypted password to be copied and used.

| Key Name | Default Value | Required |
|----------|---------------|----------|
| N/A | Selected | No |

Service Discovery Mode

This option specifies whether the connector uses the DataStax AOSS service.

- AOSS: The connector discovers Spark services via the AOSS service.
- No Service Discovery: The connector connects to Spark without using a discovery service.

| Key Name | Default Value | Required |
|----------------------|----------------------|----------|
| ServiceDiscoveryMode | No Service Discovery | No |

Service Name

The Kerberos service principal name of the Spark server.

| Key Name | Default Value | Required |
|----------------|---------------|----------|
| KrbServiceName | spark | No |

Spark Server Type

This option specifies the type of Spark server.



Note: The Shark Server 2 option is provided only for backwards compatibility with previous applications. If the connector will connect to Shark 0.9, or Spark 1.1 or later, then set Spark Thrift Server (3).

- Shark Server (1): The connector connects to a Shark Server instance.
- Shark Server 2 (2): The connector connects to a Shark Server 2 instance.
- Spark Thrift Server (3): The connector connects to a Spark Thrift Server instance.
- DFI (4): The connector connects to a DFI server instance.
- IDL (5): The connector connects to a IDL server instance.

| Key Name | Default Value | Required |
|-----------------|-------------------------|----------|
| SparkServerType | Spark Thrift Server (3) | No |

Show System Table

This option specifies whether the connector returns the spark_system table for catalog function calls such as `SQLTables` and `SQLColumns`.

- Enabled (1): The connector returns the spark_system table for catalog function calls such as `SQLTables` and `SQLColumns`.
- Disabled (0): The connector does not return the spark_system table for catalog function calls.

| Key Name | Default Value | Required |
|-----------------|---------------|----------|
| ShowSystemTable | Clear (0) | No |

Socket Timeout

The number of seconds that an operation can remain idle before it is closed.

i **Note:** This option is applicable only when asynchronous query execution is being used against Spark Thrift Server instances.

| Key Name | Default Value | Required |
|---------------|---------------|----------|
| SocketTimeout | 60 | No |

Status Endpoint Authentication

This property specifies whether the connector authenticates AOSS connections.

- No Authentication (0): The connector does not authenticate connections to AOSS endpoints.
- User Name and Password (3): The connector authenticates connections to AOSS endpoints using a user name and password.

i **Note:** This property is applicable only when AOSS dynamic service discovery is enabled.

| Key Name | Default Value | Required |
|---------------|---------------|----------|
| AOSS_AuthMech | 0 | No |

Thrift Transport

The transport protocol to use in the Thrift layer.

Select one of the following settings, or set the key to the number corresponding to the desired setting:

- Binary (0)
- SASL (1)
- HTTP (2)

i **Note:** For information about how to determine which Thrift transport protocols your Spark server supports, see [Authentication Mechanisms](#).

| Key Name | Default Value | Required |
|-----------------|---|----------|
| ThriftTransport | Binary (0) if you are connecting to Spark Server 1. SASL (1) if you are connecting to Spark Server 2. | No |

TokenCachePassPhrase

The password used to protect the token cache.

 **Note:** This option is required only on non-Windows platform.

| Key Name | Default Value | Required |
|----------------------|---------------|----------|
| TokenCachePassPhrase | None | Yes |

TokenRenewLimit

TokenRenewLimit sets the threshold (in minutes) for renewing the access token when its remaining lifetime is less than or equal to this value.

| Key Name | Default Value | Required |
|-----------------|---------------|----------|
| TokenRenewLimit | 0 | No |

Trusted Certificates

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

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

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

| Key Name | Default Value | Required |
|--------------|---|----------|
| TrustedCerts | The <code>cacerts.pem</code> file in the <code>\lib</code> 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 |

Trusted Certificates (for AOSS)

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

i **Note:** This property is applicable only when AOSS dynamic service discovery is enabled, and SSL is enabled for AOSS connections.

! **Important:**

If you are connecting from a Windows machine and the Use System Trust Store option is enabled, the connector uses the certificates from the Windows trust store instead of your specified .pem file. For more information, see [Use System Trust Store](#).

| Key Name | Default Value | Required |
|-------------------|---|----------|
| AOSS_TrustedCerts | The value being used for the non-AOSS version of this property, which defaults to the path to the <code>cacerts.pem</code> file in the <code>\lib</code> subfolder within the connector's installation directory. | No |

Two-Way SSL

This option specifies whether two-way SSL is enabled.

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

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

| Key Name | Default Value | Required |
|-----------|---------------|----------|
| TwoWaySSL | Clear (0) | No |

Unicode SQL Character Types

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

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

| Key Name | Default Value | Required |
|-----------------------------|---------------|----------|
| UseUnicodeSqlCharacterTypes | Clear (0) | No |

Use Async Exec

This option specifies whether to execute queries synchronously or asynchronously.

- Enabled (1): The connector uses an asynchronous version of the API call against Spark for executing a query.
- Disabled (0): The connector executes queries synchronously.



Note: Setting `ForceSynchronousExec` property to 1 replaces `EnableAsyncExec`.

| Key Name | Default Value | Required |
|-----------------|---------------|----------|
| EnableAsyncExec | Clear (0) | No |

Use JWT Assertion

This option specifies whether the connector uses JWT token for this connections.

- Enabled (1): The connector uses JWT token as the client credentials.
- Disabled (0): The connector does not use JWT token as the client credentials.

| Key Name | Default Value | Required |
|-----------------|---------------|----------|
| UseJWTAssertion | 0 | No |

Use Native Query

This option specifies whether the connector uses native HiveQL queries, or converts the queries emitted by an application into an equivalent form in HiveQL. If the application is Spark-aware and already emits HiveQL, then enable this option to avoid the extra overhead of query transformation.

- Enabled (1): The connector does not transform the queries emitted by an application, and executes HiveQL queries directly.
- Disabled (0): The connector transforms the queries emitted by an application and converts them into an equivalent form in HiveQL.
- Auto (2): The connector automatically sets the configuration to either 0 or 1 depending on the server's capability.



Important:

When this option is enabled, the connector cannot execute parameterized queries.

| Key Name | Default Value | Required |
|----------------|---------------|----------|
| UseNativeQuery | Clear (0) | No |

Use OIDC Discovery Endpoint

This property specifies whether to enable the OIDC discovery endpoint.

- 1: The connector enables the OIDC discovery endpoint.
- 0: The connector disables the OIDC discovery endpoint.

| Key Name | Default Value | Required |
|---------------------|---------------|----------|
| EnableOIDCDiscovery | 1 | No |

Use Only SSPI

This option specifies how the connector handles Kerberos authentication: either with the SSPI plugin or with MIT Kerberos.

- Enabled (1): The connector handles Kerberos authentication by using the SSPI plugin instead of MIT Kerberos by default.
- Disabled (0): The connector uses MIT Kerberos to handle Kerberos authentication, and only uses the SSPI plugin if the GSSAPI library is not available.

 **Important:** This option is only available in Windows.

| Key Name | Default Value | Required |
|-------------|---------------|----------|
| UseOnlySSPI | Clear (0) | No |

Use Proxy Server

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

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

| Key Name | Default Value | Required |
|----------|---------------|----------|
| UseProxy | 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 | Clear (0) | No |

Use System Trust Store (for AOSS)

This property specifies whether, when connecting to an AOSS endpoint with SSL enabled, the connector uses 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 \(for AOSS\)](#).

 **Note:** This property is available only in Windows, and applicable only when AOSS dynamic service discovery is enabled.

| Key Name | Default Value | Required |
|--------------------------|--|----------|
| AOSS_UseSystemTrustStore | The value being used for the non-AOSS version of this property, which defaults to Clear (0). | No |

User Name

The user name that you use to access Spark Thrift Server.

| Key Name | Default Value | Required |
|----------|---|---|
| UID | For User Name (2) authentication only, the default value is anonymous | Yes, if the authentication mechanism is User Name And Password (3), Windows Azure HDInsight Emulator (5), or Windows Azure HDInsight Service (6). |

| Key Name | Default Value | Required |
|----------|---------------|---|
| | | No, if the authentication mechanism is User Name (2). |

User Name (for AOSS)

The user name that you use to access the AOSS endpoints.

 **Note:** This property is applicable only when AOSS dynamic service discovery is enabled.

| Key Name | Default Value | Required |
|----------|---|----------|
| AOSS_UID | The value being used for the non-AOSS version of the User Name property (UID), which does not have a default value for User Name And Password authentication. | No |

Configuration Options Having Only Key Names

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

- [ADUserNameCase](#)
- [Async Metadata Operations](#)
- [ClusterAutostartRetry](#)
- [ClusterAutostartRetryTimeout](#)
- [ConfigsFromFileDSN](#)
- [Driver](#)
- [EnableOIDCDiscovery](#)
- [EnablePKFK](#)
- [EnableStragglerDownloadMitigation](#)
- [EnableSynchronousDownloadFallback](#)
- [EnableNativeParameterizedQuery](#)
- [EnableTelemetry](#)

- FetchResultIdleTimeout
- ForceSynchronousExec
- HTTPAuthCookies
- http.header.
- IgnoreTransactions
- MaximumStragglersPerQuery
- OAuth2RedirectUrlPort
- OAuthWebServerTimeout
- OIDCDiscoveryEndpoint
- PopulateParametersForNativeQuery
- QueryTimeoutOverride
- RateLimitRetry
- RateLimitRetryTimeout
- SSP_
- StagingAllowedLocalPaths
- StragglerDownloadMultiplier
- StragglerDownloadPadding
- StragglerDownloadQuantile
- TelemetryBatchSize
- TelemetryTimer
- ThrowOnUnsupportedPkfkRestriction
- UserAgentEntry

ADUserNameCase

This option controls whether the connector changes the user name part of an AD Kerberos UPN to all upper-case or all lower-case. The following values are supported:

- **Upper:** Change the user name to all upper-case.
- **Lower:** Change the user name to all lower-case.
- **Unchanged:** Do not modify the user name.



Note: This option is applicable only when using Active Directory Kerberos from a Windows client machine to authenticate.

| Key Name | Default Value | Required |
|----------------|---------------|----------|
| ADUserNameCase | Unchanged | No |

Async Metadata Operations

This option specifies whether the connector executes all the database metadata calls asynchronously the data source uses Thrift protocol 9.0 or later.

- **Enabled (1):** The connector executes all the database metadata calls on the Spark data source asynchronously.
- **Disabled (0):** The connector does not execute all the database metadata calls on the Spark data source asynchronously.



Note: Setting `ForceSynchronousExec` property to 1 replaces `EnableAsyncMetadata`.

| Key Name | Default Value | Required |
|---------------------|---------------|----------|
| EnableAsyncMetadata | 1 | No |

ClusterAutostartRetry

This option specifies whether the connector retries operations that receive HTTP 503 responses if the server response is returned with `Retry-After` headers.

- 1: The connector retries the operation until the time limit specified by `ClusterAutostartRetryTimeout` is exceeded. For more information, see [ClusterAutostartRetryTimeout](#).
- 0: The connector does not retry the operation, and returns an error message.

| Key Name | Default Value | Required |
|-----------------------|---------------|----------|
| ClusterAutostartRetry | 1 | No |

ClusterAutostartRetryTimeout

The number of seconds that the connector waits before stopping an attempt to retry an operation when the operation receives an HTTP 503 response with `Retry-After` headers.

See also [ClusterAutostartRetry](#).

| Key Name | Default Value | Required |
|------------------------------|---------------|----------|
| ClusterAutostartRetryTimeout | 900 | No |

ConfigsFromFileDSN

A comma-separated list of configuration names that the connector reads from the DSN file.

The connector only attempts to read the configuration values from a file DSN if the following conditions are met:

- The FILEDSN configuration is passed in via the connection string.
- The configuration key-value pairs must be inside the ODBC section in the DSN file.
- The ConfigsFromFileDSN configuration is either passed in via the connection string or via the file DSN, and the value of the ConfigsFromFileDSN configuration must contain the names, comma-separated, of the configurations to read from the DSN file.
- The value of the FILEDSN configuration is a valid directory path pointing to the location of the file DSN.



Important:

In some cases, the configuration of FILEDSN is removed from the connection string. In this case, add a FILEDSNPATH configuration to the connection string with the same value that is passed in for the FILEDSN configuration.

| Key Name | Default Value | Required |
|--------------------|---------------|----------|
| ConfigsFromFileDSN | None | No |

Driver

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

EnableOIDCDiscovery

This property specifies whether to enable the OIDC discovery.

- 1: Enables OIDC discovery.
- 0: Disables OIDC discovery.

| Key Name | Default Value | Required |
|---------------------|---------------|----------|
| EnableOIDCDiscovery | 1 | No |

EnablePKFK

This option specifies whether the connector supports SQLPrimaryKeys and SQLForeignKeys catalog functions on top of the capability of the server.

- 1: Enables the support for the SQLPrimaryKeys and SQLForeignKeys catalog functions.
- 0: Disables the support for the SQLPrimaryKeys and SQLForeignKeys catalog functions.

| Key Name | Default Value | Required |
|------------|---------------|----------|
| EnablePKFK | 1 | No |

EnableStragglerDownloadMitigation

This property specifies whether to enable the feature for mitigating straggling result file download by retrying the download.

- 1: Enables the feature for mitigating straggling result file download by retrying the download.
- 0: Disables the feature for mitigating straggling result file download.

| Key Name | Default Value | Required |
|-----------------------------------|---------------|----------|
| EnableStragglerDownloadMitigation | 0 | No |

EnableSynchronousDownloadFallback

This property specifies whether to switch from downloading result files in parallel to sequential when the number of straggling download for a query exceeds the value set for the `MaximumStragglersPerQuery` configuration.

- 1: Enables switching from downloading result file in parallel to sequential when the number of straggling download for a query exceeds the value set for the `MaximumStragglersPerQuery` configuration.
- 0: Disables switching from downloading result file in parallel to sequential even when the number of straggling download for a query exceeds the value set for the `MaximumStragglersPerQuery` configuration.



Note: This configuration is only applicable when `EnableStragglerDownloadMitigation` is set to 1.

| Key Name | Default Value | Required |
|--|---------------|----------|
| <code>EnableSynchronousDownloadFallback</code> | 0 | No |

EnableNativeParameterizedQuery

This property works only when `UseNativeQuery` is set to 1, enabling native parameterized query support.

| Key Name | Default Value | Required |
|---|---------------|----------|
| <code>EnableNativeParameterizedQuery</code> | 1 | No |

EnableTelemetry

This property specifies whether the connector enables Telemetry.

- 0: Disables telemetry.
- 1: Enables telemetry.

| Key Name | Default Value | Required |
|------------------------------|---------------|----------|
| <code>EnableTelemetry</code> | 0 | No |

FetchResultIdleTimeout

This property cancels the heartbeat thread if the fetch result idle timeout is set to a value greater than 0 (in seconds) and no fetch occurs within that time. The timeout resets after each fetch. If set to 0, the heartbeat thread does not time out.

| Key Name | Default Value | Required |
|------------------------|----------------------|----------|
| FetchResultIdleTimeout | 0 (Unset/No timeout) | No |

ForceSynchronousExec

When this option is enabled (1), the connector is forced to execute queries synchronously when connected to an HDInsight cluster.

When this option is disabled (0), the connector is able to execute queries asynchronously when connected to an HDInsight cluster.

 **Note:** This option is applicable only to HDInsight clusters.

| Key Name | Default Value | Required |
|----------------------|---------------|----------|
| ForceSynchronousExec | 0 | No |

HTTPAuthCookies

A comma-separated list of authentication cookies that are supported by the connector.

If cookie-based authentication is enabled in your server, the connector authenticates the connection once based on the provided authentication credentials. It then uses the cookie generated by the server for each subsequent request in the same connection.

| Key Name | Default Value | Required |
|-----------------|---|----------|
| HTTPAuthCookies | hive.server2.auth, JSessionID, KNOXSESSIONID, KNOX_ BACKEND-HIVE | No |

http.header.

Set a custom HTTP header by using the following syntax, where *[HeaderKey]* is the name of the header to set and *[HeaderValue]* is the value to assign to the header:

```
http.header. [HeaderKey]=[HeaderValue]
```

For example:

```
http.header.AUTHENTICATED_USER=john
```

After the connector applies the header, the `http.header.` prefix is removed from the DSN entry, leaving an entry of `[HeaderKey]=[HeaderValue]`

The example above would create the following custom HTTP header:

```
AUTHENTICATED_USER: john
```

 **Note:**

- The `http.header.` prefix is case-sensitive.
- This option is applicable only when you are using HTTP as the Thrift transport protocol. For more information, see [Thrift Transport](#).

| Key Name | Default Value | Required |
|---------------------------|---------------|----------|
| <code>http.header.</code> | None | No |

Identity_Federation_Client_Id

This property specifies the OAuth Client Id for identity federation which is used in exchanging the access token with Databricks in-house token.

| Key Name | Default Value | Required |
|---|---------------|----------|
| <code>IdentityFederationClientId</code> | None | No |

IgnoreTransactions

This property specifies whether the connector ignores transaction-related operations or returns an error.

- Enabled (1): The connector ignores any transaction related operations and returns success.
- Disabled (0): The connector returns an "operation not supported" error if it attempts to run a query that contains transaction related operations.

| Key Name | Default Value | Required |
|---------------------------------|---------------|----------|
| <code>IgnoreTransactions</code> | Clear (0) | No |

MaximumStragglersPerQuery

This property specifies the maximum number of straggling downloads to mitigate, by retrying the download, before switching from downloading result files in parallel to sequential.



Note: This configuration is only applicable when `EnableSynchronousDownloadFallback` is set to 1.

| Key Name | Default Value | Required |
|--|---------------|----------|
| <code>MaximumStragglersPerQuery</code> | 10 | No |

OAuth2RedirectUrlPort

The number of the redirect port that the connector uses for OAuth authentication.

When not specified, the connector uses 8020 as the OAuth redirect port. If 8020 is not available, the connector tries 8021, and then 8022, until the login timeout.

When assigned a single port, the connector uses it as the starting port and tries port+1, port+2, until the login timeout.

When assigned a comma separated port list, the connector tries all the ports in the list.

| Key Name | Default Value | Required |
|------------------------------------|---------------|----------|
| <code>OAuth2RedirectUrlPort</code> | None | No |

OAuthWebServerTimeout

The length of time, in seconds, for which the connector waits for a browser response during OAuth2 authentication before timing out. If set to 0, the connector waits for an indefinite amount of time.



Note: If `SQL_ATTR_LOGIN_TIMEOUT` is set, `SQL_ATTR_LOGIN_TIMEOUT` takes precedence. The connector honors `SQL_ATTR_LOGIN_TIMEOUT` when using the OAuth2 browser based authentication workflow.

| Key Name | Default Value | Required |
|------------------------------------|---------------|----------|
| <code>OAuthWebServerTimeout</code> | 120 | No |

OIDCDiscoveryEndpoint

The OIDC discovery endpoint. The connector automatically discovers `OAuth2AuthorizationEndPoint` and `OAuth2TokenEndPoint`.



Note: This property is only available when `EnableOIDCDiscovery=1`.

| Key Name | Default Value | Required |
|-----------------------|---------------|----------|
| OIDCDiscoveryEndpoint | None | No |

PopulateParametersForNativeQuery

The connector populates parameters for native parameterized queries only when `UseNativeQuery` and `EnableNativeParameterizedQuery` are set to 1.

| Key Name | Default Value | Required |
|----------------------------------|---------------|----------|
| PopulateParametersForNativeQuery | 0 | No |

QueryTimeoutOverride

The number of seconds that a query can run before it is timed out. This property overwrites the `SQL_ATTR_QUERY_TIMEOUT` attribute.



Note: When the value passed is an empty string, the connector does not attempt to override the `SQL_ATTR_QUERY_TIMEOUT` attribute.

| Key Name | Default Value | Required |
|----------------------|---------------|----------|
| QueryTimeoutOverride | 0 | No |

RateLimitRetry

This option specifies whether the connector retries operations that receive HTTP 429 responses if the server response is returned with `Retry-After` headers.

- 1: The connector retries the operation until the time limit specified by `RateLimitRetryTimeout` is exceeded. For more information, see [RateLimitRetryTimeout](#).
- 0: The connector does not retry the operation, and returns an error message.

| Key Name | Default Value | Required |
|----------------|---------------|----------|
| RateLimitRetry | 1 | No |

RateLimitRetryTimeout

The number of seconds that the connector waits before stopping an attempt to retry an operation when the operation receives an HTTP 429 response with `Retry-After` headers.

See also [RateLimitRetry](#).

| Key Name | Default Value | Required |
|-----------------------|---------------|----------|
| RateLimitRetryTimeout | 120 | No |

SSP_

Set a server-side property by using the following syntax, where *[SSPKey]* is the name of the server-side property and *[SSPValue]* is the value for that property:

```
SSP_[SSPKey]=[SSPValue]
```

After the connector applies the server-side property, the `SSP_` prefix is removed from the DSN entry, leaving an entry of `[SSPKey]=[SSPValue]`.

i **Note:**

- The `SSP_` prefix must be upper case.
- When setting a server-side property in a connection string, it is recommended that you enclose the value in braces (`{ }`) to make sure that special characters can be properly escaped.

| Key Name | Default Value | Required |
|----------|---------------|----------|
| SSP_ | None | No |

StagingAllowedLocalPaths

A comma-separated list of allowed local paths for downloading and uploading of UC Volume Ingestion files.

| Key Name | Default Value | Required |
|--------------------------|---------------|----------|
| StagingAllowedLocalPaths | None | No |

StragglerDownloadMultiplier

This property specifies the number of times that a download has to be slower than the median to be considered a straggling download. A download is considered a straggler if it takes longer to download the file than $(\text{StragglerDownloadMultiplier} * (\text{file size}/\text{median download throughput}) + \text{StragglerDownloadPadding})$.

i **Note:** This configuration is only applicable when `EnableStragglerDownloadMitigation` is set to 1

| Key Name | Default Value | Required |
|-----------------------------|---------------|----------|
| StragglerDownloadMultiplier | 1.5 | No |

StragglerDownloadPadding

This property specifies the number of seconds to give to a result file download as buffer before considering it a straggling download operation. A download is considered a straggler if it takes longer to download the file than $(\text{StragglerDownloadMultiplier} * (\text{file size}/\text{median download throughput}) + \text{StragglerDownloadPadding})$.



Note: This configuration is only applicable when `EnableStragglerDownloadMitigation` is set to 1.

| Key Name | Default Value | Required |
|--------------------------|---------------|----------|
| StragglerDownloadPadding | 5 | No |

StragglerDownloadQuantile

This property specifies the minimum fraction of result files within a batch that has to be successfully downloaded before the driver starts mitigating straggling downloads.



Note: This configuration is only applicable when `EnableStragglerDownloadMitigation` is set to 1.

| Key Name | Default Value | Required |
|---------------------------|---------------|----------|
| StragglerDownloadQuantile | 0.6 | No |

TelemetryBatchSize

This property specifies the telemetry's log batch size, when the log in the log batch hits the size, the connector sends the batch to the server.

| Key Name | Default Value | Required |
|--------------------|---------------|----------|
| TelemetryBatchSize | 200 | No |

TelemetryTimer

This property specifies the timer in seconds to indicate when the connector sends the telemetry to the server.

| Key Name | Default Value | Required |
|----------------|---------------|----------|
| TelemetryTimer | 120 | No |

ThrowOnUnsupportedPkFkRestriction

This option specifies how the connector supports SQLForeignKeys for this restriction combination: the catalog, schema and table name parameters are specified for the primary key table while those for the

foreign key table are left as NULL.

- 1: Disables the support for this restriction combination.
- 0: Enables the support for this restriction combination.

| Key Name | Default Value | Required |
|-----------------------------------|---------------|----------|
| ThrowOnUnsupportedPkFkRestriction | 0 | No |

UserAgentEntry

The User-Agent entry to be included in the HTTP request. This value is in the following format:

[ProductName]/[ProductVersion] [Comment]

Where:

- *[ProductName]* is the name of the application, with no spaces.
- *[ProductVersion]* is the version number of the application.
- *[Comment]* is an optional comment. Nested comments are not supported.

Only one User-Agent entry may be included.

| Key Name | Default Value | Required |
|----------------|---------------|----------|
| UserAgentEntry | None | No |

Third-Party Trademarks

IBM and AIX are trademarks or registered trademarks of IBM Corporation or its subsidiaries in Canada, United States, and/or other countries.

Kerberos is a trademark of the Massachusetts Institute of Technology (MIT).

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 Azure, 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.

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

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

Apache Spark, Apache, and Spark are trademarks or registered trademarks of The Apache Software Foundation or its subsidiaries in Canada, United States and/or other countries.

All other trademarks are trademarks of their respective owners.