Magnitude Simba Apache Spark ODBC Data Connector

Installation and Configuration Guide

Version 2.6.19

October 1, 2021

Copyright © 2021 Magnitude Software, Inc. 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 Magnitude.

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

Any Magnitude product described herein is licensed exclusively subject to the conditions set forth in your Magnitude 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

Magnitude Software, Inc.

www.magnitude.com

About This Guide

Purpose

The *Magnitude Simba Apache Spark ODBC Data Connector Installation and Configuration Guide* explains how to install and configure the Magnitude 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 are used when referring to book and document titles.

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

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

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

! Important:

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

Table of Contents

About the Simba Apache Spark ODBC Connector 7

Windows Connector 8
- Windows System Requirements 8
- Installing the Connector on Windows 8
- Creating a Data Source Name on Windows 9
- Configuring a DSN-less Connection on Windows 12
- Configuring AOSS Options on Windows 15
- Configuring Authentication on Windows 17
- Configuring Advanced Options on Windows 23
- Configuring a Proxy Connection on Windows 26
- Configuring HTTP Options on Windows 26
- Configuring SSL Verification on Windows 27
- Configuring Server-Side Properties on Windows 29
- Configuring Logging Options on Windows 30
- Configuring Kerberos Authentication for Windows 33
- Verifying the Connector Version Number on Windows 37

macOS Connector 39
- macOS System Requirements 39
- Installing the Connector on macOS 39
- Verifying the Connector Version Number on macOS 40

Linux Connector 41
- Linux System Requirements 41
- Installing the Connector Using the RPM File 42
- Installing the Connector Using the Tarball Package 43
- Installing the Connector on Debian 44
- Verifying the Connector Version Number on 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 on Non-Windows Machines 51
Specifying ODBC Driver Managers on Non-Windows Machines 51
Specifying the Locations of the Connector Configuration Files 52
Configuring ODBC Connections on a Non-Windows Machine 54
Creating a Data Source Name on a Non-Windows Machine 54
Configuring a DSN-less Connection on a Non-Windows Machine 58
Configuring DataStax AOSS Service Discovery on a Non-Windows Machine 60
Configuring Authentication on a Non-Windows Machine 61
Configuring SSL Verification on a Non-Windows Machine 66
Configuring Server-Side Properties on a Non-Windows Machine 67
Configuring Logging Options on a Non-Windows Machine 68
Setting Connector-Wide Configuration Options on a Non-Windows Machine 70
Testing the Connection on a Non-Windows Machine 70
Authentication Mechanisms 73
Shark Server 73
Using a Connection String 75
DSN Connection String Example 75
DSN-less Connection String Examples 75
Features 80
SQL Connector for HiveQL 80
Data Types 81
Timestamp Function Support 82
Catalog and Schema Support 82
spark_system Table 82
Server-Side Properties 83
Get Tables With Query 83
Active Directory 83
Write-back 84
Dynamic Service Discovery using DataStax AOSS 84
Security and Authentication 84
Connector Configuration Options 86
Configuration Options Appearing in the User Interface 86
Configuration Options Having Only Key Names 122
Third-Party Trademarks 128

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 on page 80.

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 *Data Access Standards* on the Simba Technologies website: https://www.simba.com/resources/data-access-standards-glossary. For complete information about the ODBC specification, see the *ODBC API Reference* from the Microsoft documentation: https://docs.microsoft.com/en-us/sql/odbc/reference/syntax/odbc-api-reference.

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

The *Installation and Configuration Guide* is suitable for users who are looking to access 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

Windows System Requirements

The Simba Apache Spark ODBC Connector supports Apache Spark versions 1.6, 2.1 through 2.4, and 3.0.

> **Note:**
>
> Support for Spark 1.6, 2.1, and 2.2 is deprecated, and will be removed in a future release of this connector. For more information, see the release notes.

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

- Administrator rights on your machine.
- A machine that meets the following system requirements:
  - One of the following operating systems:
    - Windows 10 or 8.1
    - Windows Server 2019, 2016, or 2012
  - 100 MB of available disk space

Before the connector can be used, the Visual C++ Redistributable for Visual Studio 2015 with the same bitness as the connector must also be installed. If you obtained the connector from the Simba website, then your installation of the connector automatically includes this dependency. Otherwise, you must install the redistributable manually. You can download the installation packages for the redistributable at https://www.microsoft.com/en-ca/download/details.aspx?id=48145.

Installing the Connector on Windows

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

On 64-bit Windows operating systems, 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:

- `SimbaSparkODBC32.msi` for 32-bit applications
- `SimbaSparkODBC64.msi` for 64-bit applications

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

To install the Simba Apache Spark ODBC Connector on Windows:

1. Depending on the bitness of your client application, double-click to run **SimbaSparkODBC32.msi** or **SimbaSparkODBC64.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 on 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 on page 75. For information about connector-wide settings, see Configuring a DSN-less Connection on Windows on page 12.

To create a Data Source Name on 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 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 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:
     a. From the **Service Discovery Mode** drop-down list, select **No Service Discovery**.
     b. In the **Host(s)** field, type the IP address or host name of the Spark server.
     c. 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:

a. From the **Service Discovery Mode** drop-down list, select **AOSS**.
b. 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]
```

c. 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 on Windows on page 15.

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 on Windows on page 17.

> **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 on page 73.

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 on page 73.

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 on Windows on page 26.
15. To configure the connector to connect to Spark through a proxy server, click **Proxy Options**. For more information, see Configuring a Proxy Connection on Windows on page 26.
16. To configure client-server verification over SSL, click **SSL Options**. For more information, see Configuring SSL Verification on Windows on page 27.

> **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 on Windows on page 23.
18. To configure server-side properties, click **Advanced Options** and then click **Server Side Properties**. For more information, see Configuring Server-Side Properties on Windows on page 29.
19. To configure logging behavior for the connector, click **Logging Options**. For more information, see Configuring Logging Options on Windows on page 30.
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**.

Configuring a DSN-less Connection on 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 on page 75.

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:
   - If you are using Windows 7 or earlier, click **Start** > **All Programs** > **Simba Apache Spark ODBC Connector 2.6** > **Driver Configuration**.
   - Or, if you are using Windows 8 or later, click the arrow button at the bottom of the Start screen, and then click **Simba Apache Spark ODBC Connector 2.6** > **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**.
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:
  a. From the **Service Discovery Mode** drop-down list, select **AOSS**.
  b. 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 on Windows on page 15.

5. In the Authentication area, configure authentication as needed. For more information, see Configuring Authentication on Windows on page 17.

> **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 on page 73.

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 on page 73.

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 on Windows on page 26.
9. To configure the connector to connect to Spark through a proxy server, click **Proxy Options**. For more information, see Configuring a Proxy Connection on Windows on page 26.

10. To configure client-server verification over SSL, click **SSL Options**. For more information, see Configuring SSL Verification on Windows on page 27.

> **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 on Windows on page 23.
12. To configure server-side properties, click **Advanced Options** and then click **Server Side Properties**. For more information, see Configuring Server-Side Properties on Windows on page 29.
13. To configure logging behavior for the connector, click **Logging Options**. For more information, see Configuring Logging Options on Windows on page 30.
14. To save your settings and close the Simba Spark ODBC Driver Configuration tool, click **OK**.

Configuring AOSS Options on 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 on 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**.

   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 on 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 on page 73.

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 on page 124.

Note:

On 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 Spark ODBC Driver 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 on Windows on page 27.
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 on page 33. For information about setting the Use Only SSPI advanced option, see Configuring Advanced Options on Windows on page 23.

> **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 Spark ODBC Driver 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 on Windows on page 27.
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 Spark ODBC Driver 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 Spark ODBC Driver 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 on Windows on page 27.
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 on Windows Azure HDInsight Emulator.

To configure a connection to a Spark server on 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 on 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 Spark ODBC Driver 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 abpve.
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 on Windows on page 26.

7. Click **SSL Options** and configure SSL settings as needed. For more information, see Configuring SSL Verification on Windows on page 27.
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

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 Providing a New Access Token on page 23.

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.

To configure OAuth 2.0 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**.

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.

   > **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.

Configuring Advanced Options on 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 on 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 Spark ODBC Driver 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 **Enable Auto Reconnect**.
12. In the **Rows Fetched Per Block** field, type the number of rows to be fetched per block.
13. 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).

14. In the **Default String Column Length** field, type the maximum data length for STRING columns.
15. In the **Binary Column Length** field, type the maximum data length for BINARY columns.
16. In the **Decimal Column Scale** field, type the maximum number of digits to the right of the decimal point for numeric data types.
17. 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.

18. 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.

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

Configuring a Proxy Connection on 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 on 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. In the **Proxy Username** field, type your user name for accessing the proxy server.
6. In the **Proxy Password** field, type the password corresponding to the user name.
7. 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**.
8. To save your settings and close the HTTP Proxy Options dialog box, click **OK**.

Configuring HTTP Options on 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 on page 73.

The following instructions describe how to configure HTTP 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 HTTP options on 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 on 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 on 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 Spark ODBC Driver 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. From the **Minimum TLS Version** drop-down list, select the minimum version of TLS to use when connecting to your data store.
7. To configure two-way SSL verification, select the **Two-Way SSL** check box and then do the following:
   a. In the **Client Certificate File** field, specify the full path of the PEM file containing the client's certificate.
   b. In the **Client Private Key File** field, specify the full path of the file containing the client's private key.
   c. If the private key file is protected with a password, type the password in the **Client Private Key Password** field.

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

   d. To encrypt your credentials, 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**.

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

Configuring Server-Side Properties on 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 on 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 Spark ODBC Driver 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.

   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 on Windows

To help troubleshoot issues, you can enable logging. In addition to functionality provided in the 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 on page 32.

To enable connector-wide logging on Windows:

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

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

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

   Note:

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

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

   Note:

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

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

The Simba 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 on Windows:

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

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 on Windows on page 30. 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 on 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 on page 86.
   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 on Windows. There are two prerequisites for using Active Directory Kerberos on 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:
   - If you are using Windows 7 or earlier, click **Start**, then right-click **Computer**, and then click **Properties**.

- Or, if you are using Windows 8 or later, 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:
   - If you are using Windows 7 or earlier, click **Start**, then right-click **Computer**, and then click **Properties**.
   - Or, if you are using Windows 8 or later, 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:

1. Open a command prompt:
   - If you are using Windows 7 or earlier, click **Start**, then click **All Programs**, then click **Accessories**, and then click **Command Prompt**.
   - If you are using Windows 8 or later, 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 -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`.

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.

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:
   - If you are using Windows 7 or earlier, click **Start**, then click **All Programs**, then click **Accessories**, and then click **Command Prompt**.
   - If you are using Windows 8 or later, 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 on 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 on 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

macOS System Requirements

The Simba Apache Spark ODBC Connector supports Apache Spark versions 1.6, 2.1 through 2.4, and 3.0.

> **Note:**
>
> Support for Spark 1.6, 2.1, and 2.2 is deprecated, and will be removed in a future release of this connector. For more information, see the release notes.

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

- One of the following macOS versions:
  - macOS 10.13
  - macOS 10.14
  - macOS 10.15
- 100MB of available disk space
- One of the following ODBC driver managers installed:
  - iODBC 3.52.9 or later
  - unixODBC 2.2.14 or later

Installing the Connector on 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 both 32- and 64-bit client applications.

To install the Simba Apache Spark ODBC Connector on macOS:

1. Double-click **SimbaSparkODBC.dmg** to mount the disk image.
2. Double-click **SimbaSparkODBC.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 on Non-Windows Machines on page 51.

Verifying the Connector Version Number on 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 on macOS:

- At the Terminal, run the following 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

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 1.6, 2.1 through 2.4, and 3.0.

Note:

Support for Spark 1.6, 2.1, and 2.2 is deprecated, and will be removed in a future release of this connector. For more information, see the release notes.

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

- One of the following distributions:
  - Red Hat® Enterprise Linux® (RHEL) 7
  - CentOS 7
  - SUSE Linux Enterprise Server (SLES) 11 or 12 or 15
  - Debian 8 or 9
  - Ubuntu 16.04, 18.04, or 20.04
  - Oracle Linux 7.5 or 7.6
- 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
- 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

> **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 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 on Non-Windows Machines on page 51.

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 on Non-Windows Machines on page 51.

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, 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:

- `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 on Non-Windows Machines on page 51.

Verifying the Connector Version Number on 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 on 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 on 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

AIX System Requirements

The Simba Apache Spark ODBC Connector supports Apache Spark versions 1.6, 2.1 through 2.4, and 3.0.

> **Note:**
>
> Support for Spark 1.6, 2.1, and 2.2 is deprecated, and will be removed in a future release of this connector. For more information, see the release notes.

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

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 on Non-Windows Machines on page 51.

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 1.6, 2.1 through 2.4, and 3.0.

> **Note:**
>
> Support for Spark 1.6, 2.1, and 2.2 is deprecated, and will be removed in a future release of this connector. For more information, see the release notes.

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

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 on Non-Windows Machines on page 51.

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 on 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 on Non-Windows Machines on page 51.
- 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 on page 52.

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

Specifying ODBC Driver Managers on Non-Windows Machines

You need to make sure that your machine uses the correct ODBC driver manager to load the 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 ODBCSYSINI=/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 on a 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 on page 54
- Configuring a DSN-less Connection on a Non-Windows Machine on page 58
- Configuring DataStax AOSS Service Discovery on a Non-Windows Machine on page 60
- Configuring Authentication on a Non-Windows Machine on page 61
- Configuring SSL Verification on a Non-Windows Machine on page 66
- Configuring Server-Side Properties on a Non-Windows Machine on page 67
- Configuring Logging Options on a Non-Windows Machine on page 68
- Setting Connector-Wide Configuration Options on a Non-Windows Machine on page 70
- Testing the Connection on a Non-Windows Machine on page 70

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 on a Non-Windows Machine on page 58 and Using a Connection String on page 75. For information about connector-wide settings, see

Setting Connector-Wide Configuration Options on a Non-Windows Machine on page 70.

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_
      sbu.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:
     a. Set the `ServiceDiscoveryMode` property to `No Service Discovery`.
     b. Set the `Host` property to the IP address or host name of the Spark server.
     c. 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 on page 60.

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 on page 61.

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 on a Non-Windows Machine on page 66.

> **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 on page 67.

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 on page 86.

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 on page 52.

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_sbu.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 on 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_
      sbu.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 `odbcinst.ini` configuration file.

> **Note:**
>
> If you are storing this file in its default location in the home directory, then prefix the file name with a period (`.`) so that the file becomes hidden. If you are storing this file in another location, then save it as a non-hidden file (without the prefix), and make sure that the ODBCINSTINI or ODBCSYSINI environment variable specifies the location. For more information, see Specifying the Locations of the Connector Configuration Files on page 52.

For example, the following is an `odbcinst.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_sbu.dylib
```

As another example, the following is an `odbcinst.ini` configuration file for both the 32- and 64-bit connectors on 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 on page 75.

For instructions about configuring specific connection features, see the following:

- Configuring DataStax AOSS Service Discovery on a Non-Windows Machine on page 60
- Configuring Authentication on a Non-Windows Machine on page 61
- Configuring SSL Verification on a Non-Windows Machine on page 66
- Configuring Server-Side Properties on a Non-Windows Machine on page 67

For detailed information about all the connection properties that the connector supports, see Connector Configuration Options on page 86.

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 on page 73.

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 on page 86.

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 on page 124.

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 `0`.
2. If the Spark server is configured to use SSL, then configure SSL for the connection. For more information, see Configuring SSL Verification on a Non-Windows Machine on page 66.

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 on a Non-Windows Machine on page 66.

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 on a Non-Windows Machine on page 66.

Using Windows Azure HDInsight Emulator

This authentication mechanism is available only for Spark Thrift Server instances running on 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 on 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. on page 125.

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 on 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. on page 125.
6. Configure SSL settings as needed. For more information, see Configuring SSL Verification on a Non-Windows Machine on page 66.
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

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 on page 65.

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.

To configure OAuth 2.0 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.

Configuring SSL Verification on 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.

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

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 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.

   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`.

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`.

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`.

Configuring Logging Options on 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.

LogLevel Value	Description
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 on page 86.
3. Save the `simba.sparkodbc.ini` configuration file.

Testing the Connection on a Non-Windows Machine

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

Using the iODBC Driver Manager

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

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

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

To test your connection using the iODBC driver manager:

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

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

Using the unixODBC Driver Manager

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

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

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

To test your connection using the unixODBC driver manager:

- Run isql or iusql by using the corresponding syntax:
  - `isql` *`[DataSourceName]`*
  - `iusql` *`[DataSourceName]`*

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

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

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

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 on Windows Azure, then you must use the Windows Azure HDInsight Emulator mechanism.

If you are connecting to HDInsight Service running on 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:

hive.server2.authentication	Authentication Mechanism
NOSASL	No Authentication
KERBEROS	Kerberos
NONE	User Name
LDAP	User Name and Password
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 on Windows on page 17.

For detailed instructions on how to configure authentication when using a non-Windows connector, see Configuring Authentication on a Non-Windows Machine on page 61.

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 on page 86.

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.

- *[Realm]* is the Kerberos realm of the Spark server host.
- *[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.
- *[ServiceName]* is the Kerberos service principal name of 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 on Windows Azure HDInsight Emulator

The following is the format of a DSN-less connection string that connects to a Spark Thrift Server instance running on 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 on Windows Azure HDInsight Service

The following is the format of a DSN-less connection string that connects to a Spark Thrift Server instance running on 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.

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

```
[AccessToken];ThriftTransport=2;
```

For example:

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

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;
```

Features

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

- SQL Connector for HiveQL on page 80
- Data Types on page 81
- Timestamp Function Support on page 82
- Catalog and Schema Support on page 82
- spark_system Table on page 82
- Server-Side Properties on page 83
- Get Tables With Query on page 83
- Active Directory on page 83
- Write-back on page 84
- Dynamic Service Discovery using DataStax AOSS on page 84
- Security and Authentication on page 84

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
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. Since Spark only organizes tables into schemas/databases, the connector provides a synthetic catalog named SPARK under which all of the schemas/databases are organized. The connector also maps the ODBC schema to the Spark schema/database.

Note:

When connecting to a server that supports multiple catalogs, 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.

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 Spark ODBC Driver 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 on Windows on page 29. 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 on page 67.

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 on Windows. There are two prerequisites for using Active Directory Kerberos on 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 on Windows on page 9
- Configuring a DSN-less Connection on Windows on page 12
- Configuring AOSS Options on Windows on page 15

For information about configuring this feature in a non-Windows connector, see Configuring DataStax AOSS Service Discovery on a Non-Windows Machine on page 60.

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, your Spark user name only, or your Spark user name and password. You can also authenticate a connection to an HDInsight distribution on 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 on page 73. For detailed connector configuration instructions, see Configuring Authentication on Windows on page 17 or Configuring Authentication on a Non-Windows Machine on page 61.

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 on Windows on page 27 or Configuring SSL Verification on a Non-Windows Machine on page 66.

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 Spark ODBC Driver 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:

- Access Token on page 88
- Allow Common Name Host Name
- Log Level on page 103
- Log Path on page 104

Mismatch on page 88
- Allow Common Name Host Name Mismatch (for AOSS) on page 89
- Allow Self-Signed Server Certificate on page 89
- Allow Self-Signed Server Certificate (for AOSS) on page 90
- Apply Properties with Queries on page 91
- Async Exec Poll Interval on page 91
- Authentication Flow on page 92
- Binary Column Length on page 92
- Canonicalize Principal FQDN on page 92
- CheckCertificate Revocation on page 93
- Check Certificate Revocation (for AOSS) on page 94
- Client Certificate File on page 94
- Client Private Key File on page 95
- Client Private Key Password on page 95
- Convert Key Name to Lower Case on page 95
- Database on page 96
- Decimal Column Scale on page 96
- Default String Column Length on page 97
- Delegate Kerberos Credentials on page 97
- Delegation UID on page 97
- Driver Config Take Precedence on page 98
- Enable Auto Reconnect on page 98
- Enable SSL on page 98
- Max Bytes Per Fetch Request on page 104
- Max File Size on page 105
- Max Number Files on page 105
- Mechanism on page 106
- Minimum TLS Version on page 106
- Minimum TLS Version (for AOSS) on page 107
- Password on page 108
- Password (for AOSS) on page 108
- Port on page 109
- Proxy Host on page 109
- Proxy Password on page 110
- Proxy Port on page 109
- Proxy Username on page 110
- Realm on page 110
- Rows Fetched Per Block on page 111
- Save Password (Encrypted) on page 111
- Service Discovery Mode on page 111
- Service Name on page 112
- Show System Table on page 112
- Socket Timeout on page 112
- Spark Server Type on page 113
- Status Endpoint Authentication on page 113
- Thrift Transport on page 114
- Trusted Certificates on page 115
- Trusted Certificates (for AOSS) on page 116
- Two-Way SSL on page 116
- Unicode SQL Character Types on page 117
- Use Async Exec on page 117

- Enable SSL (for AOSS) on page 99
- Fast SQLPrepare on page 100
- Get Tables With Query on page 100
- Host(s) on page 101
- Host FQDN on page 101
- HTTP Path on page 102
- Invalid Session Auto Recover on page 102
- Use Native Query on page 118
- Use Only SSPI on page 118
- Use Proxy Server on page 119
- Use System Trust Store on page 119
- Use System Trust Store (for AOSS) on page 120
- User Name on page 121
- User Name (for AOSS) on page 121

Access Token

Key Name	Default Value	Required
`Auth_AccessToken`	None	Yes, if the authentication mechanism is OAuth 2.0 (`11`).

Description

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

Allow Common Name Host Name Mismatch

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

Description

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

- 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.

Note:

This setting is applicable only when SSL is enabled.

Allow Common Name Host Name Mismatch (for AOSS)

Key Name	Default Value	Required
AOSS_ AllowHostNameCNMismatch	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 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.

Note:

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

Allow Self-Signed Server Certificate

Key Name	Default Value	Required
AllowSelfSigned ServerCert	Clear (0)	No

Description

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.

Note:

This setting is applicable only when SSL is enabled.

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

Key Name	Default Value	Required
ApplySSPWithQueries	Selected (1)	No

Description

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.

Async Exec Poll Interval

Key Name	Default Value	Required
AsyncExecPollInterval	100	No

Description

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.

Authentication Flow

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

Description

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 on page 88.

Binary Column Length

Key Name	Default Value	Required
BinaryColumnLength	32767	No

Description

The maximum data length for BINARY columns.

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

Canonicalize Principal FQDN

Key Name	Default Value	Required
ServicePrincipal Canonicalization	Selected (1)	No

Description

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.
- This option can only be disabled if the Kerberos Realm or `KrbRealm` key is specified.

CheckCertificate Revocation

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

Description

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 on page 119).

- 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 property is disabled when the `AllowSelfSignedServerCert` property is set to `1`.

Note:

This option is only available on Windows.

Check Certificate Revocation (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

Description

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) on page 120).

Client Certificate File

Key Name	Default Value	Required
ClientCert	None	No

Description

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.

Client Private Key File

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

Description

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 on page 95.

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

Client Private Key Password

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.

Description

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

Convert Key Name to Lower Case

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

Description

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.

Database

Key Name	Default Value	Required
`Schema`	`default`	No

Description

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`.

Decimal Column Scale

Key Name	Default Value	Required
`DecimalColumnScale`	`10`	`No`

Description

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

Default String Column Length

Key Name	Default Value	Required
`Default StringColumnLength`	`255`	No

Description

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.

Delegate Kerberos Credentials

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

Description

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`).

Delegation UID

Key Name	Default Value	Required
`DelegationUID`	None	No

Description

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.

Driver Config Take Precedence

Key Name	Default Value	Required
DriverConfigTakePrecedence	Clear (0)	No

Description

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.

Enable Auto Reconnect

Key Name	Default Value	Required
AutoReconnect	Selected (1)	Yes

Description

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.

Enable SSL

Key Name	Default Value	Required
SSL	Clear (0)	No

Description

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 Spark server that supports SSL.
- If you selected User Name or Windows Azure HDInsight Emulator as the authentication mechanism, SSL is not available.

Enable SSL (for AOSS)

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

Description

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.

Fast SQLPrepare

Key Name	Default Value	Required
FastSQLPrepare	Clear (0)	No

Description

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.

Get Tables With Query

Key Name	Default Value	Required
GetTablesWithQuery	0	No

Description

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.

Host(s)

Key Name	Default Value	Required
Host	None	Yes

Description

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]
```

Host FQDN

Key Name	Default Value	Required
KrbHostFQDN	_HOST	No

Description

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.

HTTP Path

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

Description

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.

Invalid Session Auto Recover

Key Name	Default Value	Required
InvalidSessionAutoRecover	Selected (1)	No

Description

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.

Log Level

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

Description

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.

Log Path

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

Description

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.

Max Bytes Per Fetch Request

Key Name	Default Value	Required
MaxBytesPerFetchRequest	300 MB	No

Description

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.

Max File Size

Key Name	Default Value	Required
LogFileSize	20971520	No

Description

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.

Max Number Files

Key Name	Default Value	Required
LogFileCount	50	No

Description

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

! Important:

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

Mechanism

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

Description

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)
- Windows Azure HDInsight Emulator (5)
- Windows Azure HDInsight Service (6)
- OAuth 2.0 (11)

Minimum TLS Version

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

Description

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.

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.

Password

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).

Description

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

Password (for AOSS)

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

Description

The password corresponding to the user name that you provided using the property User Name (for AOSS) on page 121.

Note:

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

Port

Key Name	Default Value	Required
Port	• non-HDInsight clusters: 10000 • Windows Azure HDInsight Emulator: 10001 • Windows Azure HDInsight Service: 443	Yes, if Service Discovery Mode is disabled.

Description

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

Note:

If Service Discovery Mode is enabled, this property is not used. Instead, the connector checks the Host property for information about the AOSS endpoint to connect to. For more information, see Host(s) on page 101.

Proxy Host

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

Description

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

Proxy Port

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

Description

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

Proxy Password

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

Description

The password that you use to access the proxy server.

Proxy Username

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

Description

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

Realm

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

Description

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.

Rows Fetched Per Block

Key Name	Default Value	Required
RowsFetchedPerBlock	10000	No

Description

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.

Save Password (Encrypted)

Key Name	Default Value	Required
N/A	Selected	No

Description

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.

This option is available only in the Windows connector. It appears in the Simba Spark ODBC Connector DSN Setup dialog box and the SSL Options dialog box.

! Important:

The password is obscured (not saved in plain text). However, it is still possible for the encrypted password to be copied and used.

Service Discovery Mode

Key Name	Default Value	Required
ServiceDiscoveryMode	No Service Discovery	No

Description

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.

Service Name

Key Name	Default Value	Required
`KrbServiceName`	`spark`	No

Description

The Kerberos service principal name of the Spark server.

Show System Table

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

Description

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.

Socket Timeout

Key Name	Default Value	Required
`SocketTimeout`	`60`	No

Description

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.

Spark Server Type

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

Description

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.

Status Endpoint Authentication

Key Name	Default Value	Required
AOSS_AuthMech	0	No

Description

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.

Note:

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

Thrift Transport

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

Description

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)

Note:

For information about how to determine which Thrift transport protocols your Spark server supports, see Authentication Mechanisms on page 73.

Trusted Certificates

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

Description

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.

Trusted Certificates (for AOSS)

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 `cacerts.pem` file in the `\lib` subfolder within the connector's installation directory.	No

Description

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

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 on page 119.

Two-Way SSL

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

Description

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 on page 94, Client Private Key File on page 95, and Client Private Key Password on page 95.
- 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 on page 98.

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 on page 98.

Unicode SQL Character Types

Key Name	Default Value	Required
UseUnicodeSqlCharacterTypes	Clear (0)	No

Description

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.

Use Async Exec

Key Name	Default Value	Required
EnableAsyncExec	Clear (0)	No

Description

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.

Use Native Query

Key Name	Default Value	Required
UseNativeQuery	Clear (0)	No

Description

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.

Use Only SSPI

Key Name	Default Value	Required
UseOnlySSPI	Clear (0)	No

Description

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 on Windows.

Use Proxy Server

Key Name	Default Value	Required
UseProxy	Clear (0)	No

Description

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.

Note:

This option is only available on Windows.

Use System Trust Store

Key Name	Default Value	Required
UseSystemTrustStore	Clear (0)	No

Description

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 on page 115.

Note:

This option is only available on Windows.

Use System Trust Store (for AOSS)

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

Description

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) on page 116.

Note:

This property is available only on Windows, and applicable only when AOSS dynamic service discovery is enabled.

User Name

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). No, if the authentication mechanism is User Name (2).

Description

The user name that you use to access Spark Thrift Server.

User Name (for AOSS)

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

Description

The user name that you use to access the AOSS endpoints.

Note:

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

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 on page 122
- ClusterAutostartRetry on page 123
- ClusterAutostartRetryTimeout on page 123
- Driver on page 123
- ForceSynchronousExec on page 124
- HTTPAuthCookies on page 124
- http.header. on page 125
- RateLimitRetry on page 125
- RateLimitRetryTimeout on page 126
- SSP_ on page 126
- UserAgentEntry on page 127

ADUserNameCase

Key Name	Default Value	Required
ADUserNameCase	Unchanged	No

Description

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.

ClusterAutostartRetry

Key Name	Default Value	Required
ClusterAutostartRetry	1	No

Description

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 on page 123.
- `0`: The connector does not retry the operation, and returns an error message.

ClusterAutostartRetryTimeout

Key Name	Default Value	Required
ClusterAutostartRetryTimeout	900	No

Description

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 on page 123.

Driver

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

Description

On Windows, the name of the installed connector (`Simba Spark ODBC Driver;`).

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

ForceSynchronousExec

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

Description

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.

HTTPAuthCookies

Key Name	Default Value	Required
`HTTPAuthCookies`	`hive.server2.auth, JSessionID`	No

Description

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.

http.header.

Key Name	Default Value	Required
`http.header.`	None	No

Description

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 on page 114.

RateLimitRetry

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

Description

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 on page 126.
- `0`: The connector does not retry the operation, and returns an error message.

RateLimitRetryTimeout

Key Name	Default Value	Required
`RateLimitRetryTimeout`	`120`	No

Description

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 on page 125.

SSP_

Key Name	Default Value	Required
`SSP_`	None	No

Description

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]*.

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.

UserAgentEntry

Key Name	Default Value	Required
`UserAgentEntry`	None	No

Description

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.

Third-Party Trademarks

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.

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.