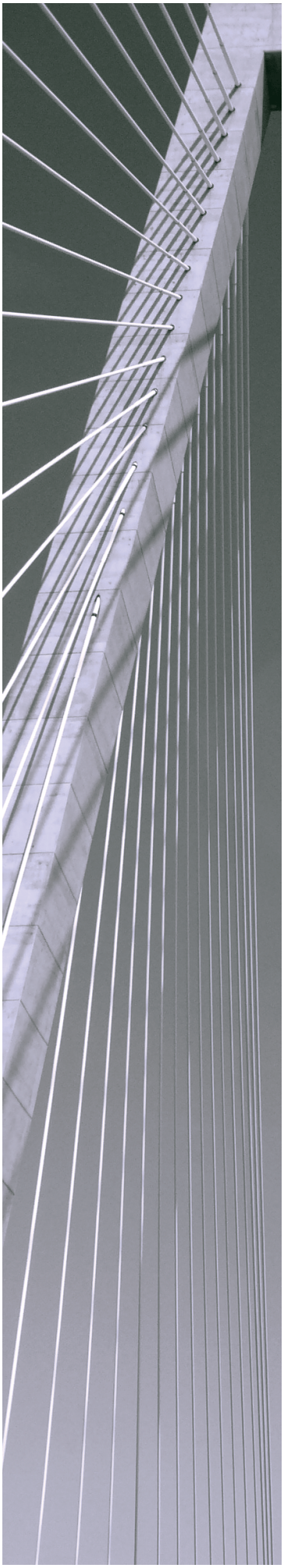

Simba Spark JDBC Driver with SQL Connector

Installation and Configuration Guide

Simba Technologies Inc.

Version 2.6.11

February 3, 2020

Copyright © 2020 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

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

Tel: +1 (604) 633-0008

Fax: +1 (604) 633-0004

www.simba.com

About This Guide

Purpose

The *Simba Spark JDBC Driver with SQL Connector Installation and Configuration Guide* explains how to install and configure the Simba Spark JDBC Driver with SQL Connector on all supported platforms. The guide also provides details related to features of the driver.

Audience

The guide is intended for end users of the Simba Spark JDBC Driver.

Knowledge Prerequisites

To use the Simba Spark JDBC Driver, the following knowledge is helpful:

- Familiarity with the platform on which you are using the Simba Spark JDBC Driver
- Ability to use the data store to which the Simba Spark JDBC Driver is connecting
- An understanding of the role of JDBC technologies in connecting to a data store
- Experience creating and configuring JDBC 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 Spark JDBC Driver 6

System Requirements 7

Simba Spark JDBC Driver Files 8

Installing and Using the Simba Spark JDBC Driver 9
- Referencing the JDBC Driver Libraries 9
- Registering the Driver Class 10
- Building the Connection URL 11

Configuring Authentication 13
- Using No Authentication 13
- Using Kerberos 13
- Using User Name 14
- Using User Name And Password (LDAP) 15
- Authentication Mechanisms 16
- Configuring Kerberos Authentication for Windows 18

Increasing the Connection Speed 24

Configuring SSL 25

Configuring Virtual Private Cloud (VPC) Services 27

Configuring Logging 29

Features 31
- SQL Query versus HiveQL Query 31
- Data Types 31
- Catalog and Schema Support 32
- Write-back 32
- Security and Authentication 32

Driver Configuration Options 34
- AllowSelfSignedCerts 34
- AsyncExecPollInterval 35
- AuthMech 35
- CAIssuedCertsMismatch 36
- CatalogSchemaSwitch 36
- DecimalColumnScale 36

DefaultStringColumnLength ... 37
DelegationUID ... 37
DnsResolver ... 37
DnsResolverArg ... 38
FastConnection ... 38
httpPath ... 38
KrbAuthType ... 39
KrbHostFQDN ... 40
KrbRealm ... 40
KrbServiceName ... 40
LogLevel ... 41
LogPath ... 42
PreparedMetaLimitZero ... 42
PWD ... 42
RowsFetchedPerBlock ... 43
ServerVersion ... 43
SocketFactory ... 44
SocketFactoryArg ... 44
SocketTimeout ... 44
SSL ... 45
SSLKeyStore ... 45
SSLKeyStorePwd ... 46
SSLTrustStore ... 46
SSLTrustStorePwd ... 47
StripCatalogName ... 47
SubjectAlternativeNamesHostNames ... 47
TransportMode ... 48
UID ... 49
UseNativeQuery ... 49

Third-Party Trademarks ... 50

About the Simba Spark JDBC Driver

The Simba Spark JDBC Driver is used for direct SQL and HiveQL access to Apache Hadoop / Spark, enabling Business Intelligence (BI), analytics, and reporting on Hadoop / Spark-based data. The driver 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 driver is configurable to pass the query through to the database for processing. The driver 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 Features on page 31.

The Simba Spark JDBC Driver complies with the JDBC 4.0, 4.1, and 4.2 data standards. JDBC 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 JDBC driver, which connects an application to the database. For more information about JDBC, see *Data Access Standards* on the Simba Technologies website: https://www.simba.com/resources/data-access-standards-glossary.

Note:

Support for JDBC 4.0 is deprecated, and will be removed in a future release of this driver. For more information, see the release notes.

This guide is suitable for users who want to access data residing within Spark from their desktop environment. Application developers might also find the information helpful. Refer to your application for details on connecting via JDBC.

System Requirements

Each machine where you use the Simba Spark JDBC Driver must have Java Runtime Environment (JRE) 7.0 or 8.0 installed. If you are using the driver with JDBC API version 4.2, then you must use JRE 8.0.

The driver supports Apache Spark versions 1.6 and 2.1 through 2.4.

! Important:

- Support for Spark 1.6, 2.1, and 2.2 is deprecated, and will be removed in a future release of this driver. For more information, see the release notes.
- The driver only supports connections to Spark Thrift Server instances. It does not support connections to Shark Server instances.

Simba Spark JDBC Driver Files

The Simba Spark JDBC Driver is delivered in the following ZIP archives, where *[Version]* is the version number of the driver:

- `SparkJDBC4_[Version].zip`
- `SparkJDBC41_[Version].zip`
- `SparkJDBC42_[Version].zip`

The archive contains the driver supporting the JDBC API version indicated in the archive name, as well as release notes and third-party license information. In addition, the required third-party libraries and dependencies are packaged and shared in the driver JAR file in the archive.

Note:

Support for JDBC 4.0 is deprecated, and will be removed in a future release of this driver. For more information, see the release notes.

Installing and Using the Simba Spark JDBC Driver

To install the Simba Spark JDBC Driver on your machine, extract the files from the appropriate ZIP archive to the directory of your choice.

! Important:

If you received a license file through email, then you must copy the file into the same directory as the driver JAR file before you can use the Simba Spark JDBC Driver.

To access a Spark data store using the Simba Spark JDBC Driver, you need to configure the following:

- The list of driver library files (see Referencing the JDBC Driver Libraries on page 9)
- The `Driver` or `DataSource` class (see Registering the Driver Class on page 10)
- The connection URL for the driver (see Building the Connection URL on page 11)

! Important:

The Simba Spark JDBC Driver provides read-only access to Spark Thrift Server instances. It does not support connections to Shark Server instances.

Referencing the JDBC Driver Libraries

Before you use the Simba Spark JDBC Driver, the JDBC application or Java code that you are using to connect to your data must be able to access the driver JAR files. In the application or code, specify all the JAR files that you extracted from the ZIP archive.

Using the Driver in a JDBC Application

Most JDBC applications provide a set of configuration options for adding a list of driver library files. Use the provided options to include all the JAR files from the ZIP archive as part of the driver configuration in the application. For more information, see the documentation for your JDBC application.

Using the Driver in Java Code

You must include all the driver library files in the class path. This is the path that the Java Runtime Environment searches for classes and other resource files. For more

information, see "Setting the Class Path" in the appropriate Java SE Documentation.

For Java SE 7:

- For Windows: http://docs.oracle.com/javase/7/docs/technotes/tools/windows/classpath.html
- For Linux and Solaris: http://docs.oracle.com/javase/7/docs/technotes/tools/solaris/classpath.html

For Java SE 8:

- For Windows: http://docs.oracle.com/javase/8/docs/technotes/tools/windows/classpath.html
- For Linux and Solaris: http://docs.oracle.com/javase/8/docs/technotes/tools/solaris/classpath.html

Registering the Driver Class

Before connecting to your data, you must register the appropriate class for your application.

The following classes are used to connect the Simba Spark JDBC Driver to Spark data stores:

- The `Driver` classes extend `java.sql.Driver`.
- The `DataSource` classes extend `javax.sql.DataSource` and `javax.sql.ConnectionPoolDataSource`.

The driver supports the following fully-qualified class names (FQCNs) that are independent of the JDBC version:

- `com.simba.spark.jdbc.Driver`
- `com.simba.spark.jdbc.DataSource`

The following sample code shows how to use the `DriverManager` to establish a connection for JDBC 4.2:

```
private static Connection connectViaDM() throws Exception
{
   Connection connection = null;
   Class.forName(DRIVER_CLASS);
   connection = DriverManager.getConnection(CONNECTION_URL);
   return connection;
}
```

The following sample code shows how to use the `DataSource` class to establish a connection:

```
private static Connection connectViaDS() throws Exception
{
   Connection connection = null;
   Class.forName(DRIVER_CLASS);
   DataSource ds = new com.simba.spark.jdbc.DataSource();
   ds.setURL(CONNECTION_URL);
   connection = ds.getConnection();
   return connection;
}
```

Building the Connection URL

Use the connection URL to supply connection information to the data store that you are accessing. The following is the format of the connection URL for the Simba Spark JDBC Driver, where *[Host]* is the DNS or IP address of the Spark server and *[Port]* is the number of the TCP port that the server uses to listen for client requests:

```
jdbc:spark://[Host]:[Port]
```

Note:

By default, Spark uses port 10000.

By default, the driver uses the schema named **default** and authenticates the connection using the user name **spark**.

You can specify optional settings such as the schema to use or any of the connection properties supported by the driver. For a list of the properties available in the driver, see Driver Configuration Options on page 34. If you specify a property that is not supported by the driver, then the driver attempts to apply the property as a Spark server-side property for the client session.

The following is the format of a connection URL that specifies some optional settings:

```
jdbc:spark://[Host]:[Port]/[Schema];[Property1]=[Value];
[Property2]=[Value];...
```

For example, to connect to port 11000 on an Spark server installed on the local machine, use a schema named default2, and authenticate the connection using a user name and password, you would use the following connection URL:

```
jdbc:spark://localhost:11000/default2;AuthMech=3;UID=simba;PWD=simba
```

Important:

- Properties are case-sensitive.
- Do not duplicate properties in the connection URL.

Note:

If you specify a schema in the connection URL, 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.

Configuring Authentication

The Simba Spark JDBC Driver supports the following authentication mechanisms:

- No Authentication
- Kerberos
- User Name
- User Name And Password

You configure the authentication mechanism that the driver uses to connect to Spark by specifying the relevant properties in the connection URL.

For information about selecting an appropriate authentication mechanism when using the Simba Spark JDBC Driver, see Authentication Mechanisms on page 16.

For information about the properties you can use in the connection URL, see Driver Configuration Options on page 34.

Note:

In addition to authentication, you can configure the driver to connect over SSL. For more information, see Configuring SSL on page 25.

Using No Authentication

You provide this information to the driver in the connection URL. For more information about the syntax of the connection URL, see Building the Connection URL on page 11.

To configure a connection without authentication:

1. Set the `AuthMech` property to `0`.
2. Set the `transportMode` property to `binary`.

For example:

```
jdbc:spark://localhost:10000;AuthMech=0;
transportMode=binary;
```

Using Kerberos

Kerberos must be installed and configured before you can use this authentication mechanism. For information about configuring and operating Kerberos on Windows, see Configuring Kerberos Authentication for Windows on page 18. For other operating

systems, see the MIT Kerberos documentation: http://web.mit.edu/kerberos/krb5-latest/doc/.

You provide this information to the driver in the connection URL. For more information about the syntax of the connection URL, see Building the Connection URL on page 11.

Note:

When you use this authentication mechanism, SASL is the only Thrift transport protocol that is supported. The driver uses SASL by default, so you do not need to set the `transportMode` property.

To configure default Kerberos authentication:

1. Set the `AuthMech` property to `1`.
2. To use the default realm defined in your Kerberos setup, do not set the `KrbRealm` property.

   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 `KrbRealm` property to the realm of the Spark server.
3. Set the `KrbHostFQDN` property to the fully qualified domain name of the Spark server host.

For example, the following connection URL connects to a Spark server with Kerberos enabled, but without SSL enabled:

```
jdbc:spark://node1.example.com:10000;AuthMech=1;
KrbRealm=EXAMPLE.COM;KrbHostFQDN=node1.example.com;
KrbServiceName=spark
```

In this example, Kerberos is enabled for JDBC connections, the Kerberos service principal name is spark/node1.example.com@EXAMPLE.COM, the host name for the data source is node1.example.com, and the server is listening on port 10000 for JDBC connections.

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.

You provide this information to the driver in the connection URL. For more information about the syntax of the connection URL, see Building the Connection URL on page 11.

To configure User Name authentication:

1. Set the `AuthMech` property to `2`.
2. Set the `transportMode` property to `sasl`.
3. Set the `UID` property to an appropriate user name for accessing the Spark server.

For example:

```
jdbc:spark://node1.example.com:10000;AuthMech=2;
transportMode=sasl;UID=spark
```

Using User Name And Password (LDAP)

This authentication mechanism requires a user name and a password. It is most commonly used with LDAP authentication.

You provide this information to the driver in the connection URL. For more information about the syntax of the connection URL, see Building the Connection URL on page 11.

To configure User Name And Password authentication:

1. Set the `AuthMech` property to `3`.
2. Set the `transportMode` property to the transport protocol that you want to use in the Thrift layer.
3. If you set the `transportMode` property to `http`, then set the `httpPath` property to the partial URL corresponding to the Spark server. Otherwise, do not set the `httpPath` property.
4. Set the `UID` property to an appropriate user name for accessing the Spark server.
5. Set the `PWD` property to the password corresponding to the user name you provided.

For example, the following connection URL connects to a Spark server with LDAP authentication enabled:

```
jdbc:spark://node1.example.com:10000;AuthMech=3;
transportMode=http;httpPath=cliservice;UID=spark;PWD=simba;
```

In this example, user name and password (LDAP) authentication is enabled for JDBC connections, the LDAP user name is spark, the password is simba, and the server is listening on port 10000 for JDBC connections.

Authentication Mechanisms

To connect to a Spark server, you must configure the Simba Spark JDBC Driver 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.

Spark Thrift Server supports the following authentication mechanisms:

- No Authentication (see Using No Authentication on page 13)
- Kerberos (see Using Kerberos on page 13)
- User Name (see Using User Name on page 14)
- User Name And Password (see Using User Name And Password (LDAP) on page 15)

Most default configurations of Spark Thrift Server require User Name authentication. If you are unable to connect to your Spark server using User Name authentication, then verify the authentication mechanism configured for your Spark server by examining the `hive-site.xml` file. Examine the following properties to determine which authentication mechanism your server is set to use:

- `hive.server2.authentication`: This property sets the authentication mode for Spark Server 2. The following values are available:
  - `NONE` enables plain SASL transport. This is the default value.
  - `NOSASL` disables the Simple Authentication and Security Layer (SASL).
  - `KERBEROS` enables Kerberos authentication and delegation token authentication.
  - `PLAINSASL` enables user name and password authentication using a cleartext password mechanism.
  - `LDAP` enables user name and password authentication using the Lightweight Directory Access Protocol (LDAP).
- `hive.server2.enable.doAs`: If this property is set to the default value of `TRUE`, then Spark processes queries as the user who submitted the query. If this property is set to `FALSE`, then queries are run as the user that runs the `hiveserver2` process.

The following table lists the authentication mechanisms to configure for the driver based on the settings in the `hive-site.xml` file.

hive.server2.authentication	hive.server2.enable.doAs	Driver Authentication Mechanism
NOSASL	FALSE	No Authentication
KERBEROS	TRUE or FALSE	Kerberos
KERBEROS	TRUE	Delegation Token
NONE	TRUE or FALSE	User Name
PLAINSASL or LDAP	TRUE or FALSE	User Name And Password

Note:

It is an error to set hive.server2.authentication to NOSASL and hive.server2.enable.doAs to true. This configuration will not prevent the service from starting up, but results in an unusable service.

For more information about authentication mechanisms, refer to the documentation for your Hadoop / Spark distribution. See also "Running Hadoop in Secure Mode" in the Apache Hadoop documentation: http://hadoop.apache.org/docs/r0.23.7/hadoop-project-dist/hadoop-common/ClusterSetup.html#Running_Hadoop_in_Secure_Mode.

Using No Authentication

When hive.server2.authentication is set to NOSASL, you must configure your connection to use No Authentication.

Using Kerberos

When connecting to a Spark Thrift Server instance and hive.server2.authentication is set to KERBEROS, you must configure your connection to use Kerberos or Delegation Token authentication.

Using User Name

When connecting to a Spark Thrift Server instance and hive.server2.authentication is set to NONE, you must configure your connection to use User Name authentication. Validation of the credentials that you include depends on hive.server2.enable.doAs:

- If `hive.server2.enable.doAs` is set to `TRUE`, then the server attempts to map the user name provided by the driver from the driver configuration to an existing operating system user on the host running Spark Thrift Server. If this user name does not exist in the operating system, then the user group lookup fails and existing HDFS permissions are used. For example, if the current user group is allowed to read and write to the location in HDFS, then read and write queries are allowed.
- If `hive.server2.enable.doAs` is set to `FALSE`, then the user name in the driver configuration is ignored.

If no user name is specified in the driver configuration, then the driver defaults to using **spark** as the user name.

Using User Name And Password

When connecting to a Spark Thrift Server instance and the server is configured to use the SASL-PLAIN authentication mechanism with a user name and a password, you must configure your connection to use User Name And Password authentication.

Configuring Kerberos Authentication for Windows

You can configure your Kerberos setup so that you use the MIT Kerberos Ticket Manager to get the Ticket Granting Ticket (TGT), or configure the setup so that you can use the driver to get the ticket directly from the Key Distribution Center (KDC). Also, if a client application obtains a Subject with a TGT, it is possible to use that Subject to authenticate the connection.

Downloading and Installing MIT Kerberos for Windows

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.

3. Follow the instructions in the installer to complete the installation process.
4. When the installation completes, click **Finish**.

Using the MIT Kerberos Ticket Manager to Get Tickets

Setting the KRB5CCNAME Environment Variable

You must set the KRB5CCNAME environment variable to your credential cache file.

To set the KRB5CCNAME environment variable:

1. Click **Start**, then right-click **Computer**, and then click **Properties**.
2. Click **Advanced System Settings**.
3. In the System Properties dialog box, on the **Advanced** tab, click **Environment Variables**.
4. In the Environment Variables dialog box, under the System Variables list, click **New**.
5. In the **New System Variable** dialog box, in the Variable Name field, type **KRB5CCNAME**.
6. In the **Variable Value** field, type the path for your credential cache file. For example, type `C:\KerberosTickets.txt`.
7. Click **OK** to save the new variable.
8. Make sure that the variable appears in the System Variables list.
9. Click **OK** to close the Environment Variables dialog box, and then click **OK** to close the System Properties dialog box.
10. Restart your machine.

Getting a Kerberos Ticket

To get a Kerberos ticket:

1. Click **Start**, then click **All Programs**, and then click the **Kerberos for Windows (64-bit)** or **Kerberos for Windows (32-bit)** program group.
2. Click **MIT Kerberos Ticket Manager**.
3. In the MIT Kerberos Ticket Manager, click **Get Ticket**.
4. 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 the MIT Kerberos Ticket Manager.

Authenticating to the Spark Server

You provide this information to the driver in the connection URL. For more information about the syntax of the connection URL, see Building the Connection URL on page 11.

To authenticate to the Spark server:

- Use a connection URL that has the following properties defined:
  - `AuthMech`
  - `KrbHostFQDN`
  - `KrbRealm`
  - `KrbServiceName`

For detailed information about these properties, see Driver Configuration Options on page 34

Using the Driver to Get Tickets

Deleting the KRB5CCNAME Environment Variable

To enable the driver to get Ticket Granting Tickets (TGTs) directly, make sure that the KRB5CCNAME environment variable has not been set.

To delete the KRB5CCNAME environment variable:

1. Click the **Start** button , then right-click **Computer**, and then click **Properties**.
2. Click **Advanced System Settings**.
3. In the System Properties dialog box, click the **Advanced** tab and then click **Environment Variables**.
4. In the Environment Variables dialog box, check if the KRB5CCNAME variable appears in the System variables list. If the variable appears in the list, then select the variable and click **Delete**.
5. Click **OK** to close the Environment Variables dialog box, and then click **OK** to close the System Properties dialog box.

Setting Up the Kerberos Configuration File

To set up the Kerberos configuration file:

1. Create a standard `krb5.ini` file and place it in the `C:\Windows` directory.
2. Make sure that the KDC and Admin server specified in the `krb5.ini` file can be resolved from your terminal. If necessary, modify `C:\Windows\System32\drivers\etc\hosts`.

Setting Up the JAAS Login Configuration File

To set up the JAAS login configuration file:

1. Create a JAAS login configuration file that specifies a keytab file and `doNotPrompt=true`.

   For example:

   ```
   Client {
   com.sun.security.auth.module.Krb5LoginModule required
   useKeyTab=true
   keyTab="PathToTheKeyTab"
   principal="simba@SIMBA"
   doNotPrompt=true;
   };
   ```

2. Set the `java.security.auth.login.config` system property to the location of the JAAS file.

   For example: `C:\KerberosLoginConfig.ini`.

Authenticating to the Spark Server

You provide this information to the driver in the connection URL. For more information about the syntax of the connection URL, see Building the Connection URL on page 11.

To authenticate to the Spark server:

- Use a connection URL that has the following properties defined:

  - `AuthMech`
  - `KrbHostFQDN`
  - `KrbRealm`
  - `KrbServiceName`

  For detailed information about these properties, see Driver Configuration Options on page 34.

Using an Existing Subject to Authenticate the Connection

If the client application obtains a Subject with a TGT, then that Subject can be used to authenticate the connection to the server.

To use an existing Subject to authenticate the connection:

1. Create a PrivilegedAction for establishing the connection to the database.

   For example:

```
// Contains logic to be executed as a privileged action
public class AuthenticateDriverAction
implements PrivilegedAction<Void>
{
// The connection, which is established as a
PrivilegedAction
Connection con;

// Define a string as the connection URL
static String ConnectionURL =
"jdbc:spark://192.168.1.1:10000";

/**
* Logic executed in this method will have access to the
* Subject that is used to "doAs". The driver will get
* the Subject and use it for establishing a connection
* with the server.
*/
@Override
public Void run()
{
try
{
// Establish a connection using the connection URL
con = DriverManager.getConnection(ConnectionURL);
}
catch (SQLException e)
{
// Handle errors that are encountered during
// interaction with the data store
e.printStackTrace();
}
catch (Exception e)
{
// Handle other errors
e.printStackTrace();
}
```

```
return null;
}
}
```

2. Run the PrivilegedAction using the existing Subject, and then use the connection.

   For example:

```
// Create the action
AuthenticateDriverAction authenticateAction = new
AuthenticateDriverAction();
// Establish the connection using the Subject for
// authentication.
Subject.doAs(loginConfig.getSubject(),
authenticateAction);
// Use the established connection.
authenticateAction.con;
```

Increasing the Connection Speed

If you want to speed up the process of connecting to the data store, you can disable several of the connection checks that the driver performs.

! Important:

Enabling these options can speed up the connection process, but may result in errors. In particular, the `ServerVersion` must match the version number of the server, otherwise errors may occur.

To increase the driver's connection speed:

1. To bypass the connection testing process, set the `FastConnection` property to `1`.
2. To bypass the server version check, set the `ServerVersion` property to the version number of the Spark server that you are connecting to.

   For example, to connect to Spark 2.4.0, set the `ServerVersion` property to `2.4.0`.

Configuring SSL

Note:

In this documentation, "SSL" indicates both TLS (Transport Layer Security) and SSL (Secure Sockets Layer). The driver supports industry-standard versions of TLS/SSL.

If you are connecting to a Spark server that has Secure Sockets Layer (SSL) enabled, you can configure the driver to connect to an SSL-enabled socket. When connecting to a server over SSL, the driver uses one-way authentication to verify the identity of the server.

One-way authentication requires a signed, trusted SSL certificate for verifying the identity of the server. You can configure the driver to access a specific TrustStore or KeyStore that contains the appropriate certificate. If you do not specify a TrustStore or KeyStore, then the driver uses the default Java TrustStore named `jssecacerts`. If `jssecacerts` is not available, then the driver uses `cacerts` instead.

You provide this information to the driver in the connection URL. For more information about the syntax of the connection URL, see Building the Connection URL on page 11.

To configure SSL:

1. Set the `SSL` property to `1`.
2. If you are not using one of the default Java TrustStores, then do one of the following:
   - Create a TrustStore and configure the driver to use it:
     a. Create a TrustStore containing your signed, trusted server certificate.
     b. Set the `SSLTrustStore` property to the full path of the TrustStore.
     c. Set the `SSLTrustStorePwd` property to the password for accessing the TrustStore.
   - Or, create a KeyStore and configure the driver to use it:
     a. Create a KeyStore containing your signed, trusted server certificate.
     b. Set the `SSLKeyStore` property to the full path of the KeyStore.
     c. Set the `SSLKeyStorePwd` property to the password for accessing the KeyStore.
3. Optionally, to allow the SSL certificate used by the server to be self-signed, set the `AllowSelfSignedCerts` property to `1`.

> **Important:**
>
> When the `AllowSelfSignedCerts` property is set to `1`, SSL verification is disabled. The driver does not verify the server certificate against the trust store, and does not verify if the server's host name matches the common name or subject alternative names in the server certificate.

4. Optionally, to allow the common name of a CA-issued certificate to not match the host name of the Spark server, set the `CAIssuedCertNamesMismatch` property to `1`.
5. If you want the driver to use the subject alternative names of the certificate instead of the common name when checking if the certificate name matches the host name of the Spark server, then set the following properties:
   a. Make sure that the `AllowSelfSignedCerts` property is set to `0`.
   b. Set the `CAIssuedCertNamesMismatch` property to `1` if you have not already done so.
   c. Set the `SubjectAlternativeNamesHostNames` property to `1`.

For example, the following connection URL connects to a data source using username and password (LDAP) authentication, with SSL enabled:

```
jdbc:spark://localhost:10000;AuthMech=3;SSL=1;
SSLKeyStore=C:\\Users\\bsmith\\Desktop\\keystore.jks;SSLKeyS
torePwd=simbaSSL123;UID=spark;PWD=simba123
```

> **Note:**
>
> For more information about the connection properties used in SSL connections, see Driver Configuration Options on page 34.

Configuring Virtual Private Cloud (VPC) Services

You can configure the driver to connect to Spark using a Virtual Private Cloud (VPC) service. To do this, use the following connection properties:

- `SocketFactory`
- `SocketFactoryArg`
- `DnsResolver`
- `DnsResolverArg`

The `SocketFactory` property extends `javax.net.SocketFactory`, and the `DnsResolver` property implements `com.interfaces.networking.CustomDnsResolver`. The `SocketFactoryArg` and `DnsResolverArg` properties pass any required arguments to these services.

The properties in the following instructions are all optional, and only need to be used if you are connecting through the relevant service.

To configure the driver to use a VPC:

1. If necessary, set the `SocketFactory` property to the fully-qualified class path for a class that extends `javax.net.SocketFactory` and provides the socket factory implementation.
2. If necessary, set the `SocketFactoryArg` to a string argument to pass to the constructor of the class indicated by the `SocketFactory` property.
3. If necessary, set the `DnsResolver` property to the fully-qualified class path for a class that extends the `DnsResolver` interface for the driver, `com.interfaces.networking.CustomDnsResolver`.
4. If necessary, set the `DnsResolverArg` to a string argument to pass to the constructor of the class indicated by the `DnsResolver` property.

For example, the following connection URLs show how to connect to data sources using the supported VPCs.

Using SocketFactory:

```
jdbc:spark://localhost:10000;UID=jsmith;SocketFactory=com.simba.junit.jdbc.utils.CustomSocketFactory;SocketFactoryArg=Args;
```

Using DnsResolver:

```
jdbc:spark://localhost:10000;UID=jsmith;DnsResolver=com.simba.junit.jdbc.utils.TestDnsResolver;DnsResolverArg=agrs;
```

Using both SocketFactory and DnsResolver:

```
jdbc:spark://localhost:10000;UID=jsmith;DnsResolver=com.simba.junit.jdbc.utils.TestDnsResolver;DnsResolverArg=Agrs;SocketFactory=com.simba.junit.jdbc.utils.CustomSocketFactory;SocketFactoryArg=Args;
```

Configuring Logging

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

! Important:

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

The settings for logging apply to every connection that uses the Simba Spark JDBC Driver, so make sure to disable the feature after you are done using it.

In the connection URL, set the `LogLevel` key to enable logging at the desired level of detail. The following table lists the logging levels provided by the Simba Spark JDBC Driver, in order from least verbose to most verbose.

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

To enable logging:

1. Set the `LogLevel` property to the desired level of information to include in log files.
2. Set the `LogPath` property to the full path to the folder where you want to save log files. To make sure that the connection URL is compatible with all JDBC applications, escape the backslashes (`\`) in your file path by typing another backslash.

For example, the following connection URL enables logging level 3 and saves the log files in the `C:\temp` folder:

```
jdbc:spark://localhost:11000;LogLevel=3;LogPath=C:\\temp
```

3. To make sure that the new settings take effect, restart your JDBC application and reconnect to the server.

The Simba Spark JDBC Driver produces the following log files in the location specified in the `LogPath` property:

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

If the `LogPath` value is invalid, then the driver sends the logged information to the standard output stream (`System.out`).

To disable logging:

1. Set the `LogLevel` property to `0`.
2. To make sure that the new setting takes effect, restart your JDBC application and reconnect to the server.

Features

More information is provided on the following features of the Simba Spark JDBC Driver:

- SQL Query versus HiveQL Query on page 31
- Data Types on page 31
- Catalog and Schema Support on page 32
- Write-back on page 32
- Security and Authentication on page 32

SQL Query versus HiveQL Query

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

Data Types

The Simba Spark JDBC Driver supports many common data formats, converting between Spark, SQL, and Java data types.

The following table lists the supported data type mappings.

Spark Type	SQL Type	Java Type
BIGINT	BIGINT	java.math.BigInteger
BINARY	VARBINARY	byte[]
BOOLEAN	BOOLEAN	Boolean
DATE	DATE	java.sql.Date
DECIMAL	DECIMAL	java.math.BigDecimal
DOUBLE	DOUBLE	Double
FLOAT	REAL	Float

Spark Type	SQL Type	Java Type
INT	INTEGER	Long
SMALLINT	SMALLINT	Integer
TIMESTAMP	TIMESTAMP	java.sql.Timestamp
TINYINT	TINYINT	Short
VARCHAR	VARCHAR	String

Catalog and Schema Support

The Simba Spark JDBC Driver supports both catalogs and schemas to make it easy for the driver to work with various JDBC applications. Since Spark only organizes tables into schemas/databases, the driver provides a synthetic catalog named SPARK under which all of the schemas/databases are organized. The driver also maps the JDBC schema to the Spark schema/database.

Note:

Setting the `CatalogSchemaSwitch` connection property to `1` will cause Spark catalogs to be treated as schemas in the driver as a restriction for filtering.

Write-back

The Simba Spark JDBC Driver 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 driver is unable to translate the statement to SQL and instead falls back to using HiveQL.

Security and Authentication

To protect data from unauthorized access, some Spark data stores require connections to be authenticated with user credentials or the SSL protocol. The Simba Spark JDBC

Driver provides full support for these authentication protocols.

Note:

In this documentation, "SSL" indicates both TLS (Transport Layer Security) and SSL (Secure Sockets Layer). The driver supports industry-standard versions of TLS/SSL.

The driver provides mechanisms that allow you to authenticate your connection using the Kerberos protocol, your Spark user name only, or your Spark user name and password. 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 16. For detailed driver configuration instructions, see Configuring Authentication on page 13.

Additionally, the driver supports SSL connections with one-way authentication. If the server has an SSL-enabled socket, then you can configure the driver to connect to it.

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 on page 25.

The SSL version that the driver supports depends on the JVM version that you are using. For information about the SSL versions that are supported by each version of Java, see "Diagnosing TLS, SSL, and HTTPS" on the Java Platform Group Product Management Blog: https://blogs.oracle.com/java-platform-group/entry/diagnosing_tls_ssl_and_https.

Note:

The SSL version used for the connection is the highest version that is supported by both the driver and the server, which is determined at connection time.

Driver Configuration Options

Driver Configuration Options lists and describes the properties that you can use to configure the behavior of the Simba Spark JDBC Driver.

You can set configuration properties using the connection URL. For more information, see Building the Connection URL on page 11.

Note:

Property names and values are case-sensitive.

AllowSelfSignedCerts

Default Value	Data Type	Required
0	Integer	No

Description

This property specifies whether the driver allows the server to use self-signed SSL certificates.

- `1`: The driver allows self-signed certificates.

  Important:

  When this property is set to `1`, SSL verification is disabled. The driver does not verify the server certificate against the trust store, and does not verify if the server's host name matches the common name or subject alternative name in the server certificate.

- `0`: The driver does not allow self-signed certificates.

Note:

This property is applicable only when SSL connections are enabled.

AsyncExecPollInterval

Default Value	Data Type	Required
10	Integer	No

Description

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

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

Note:

This option is applicable only to HDInsight clusters.

AuthMech

Default Value	Data Type	Required
Depends on the transportMode setting. For more information, see TransportMode on page 48.	Integer	No

Description

The authentication mechanism to use. Set the property to one of the following values:

- 0 for No Authentication.
- 1 for Kerberos.
- 2 for User Name.
- 3 for User Name And Password.

CAIssuedCertsMismatch

Default Value	Data Type	Required
0	Integer	No

Description

This property specifies whether the driver requires the name of the CA-issued SSL certificate to match the host name of the Spark server.

- 0: The driver requires the names to match.
- 1: The driver allows the names to mismatch.

Note:

This property is applicable only when SSL connections are enabled.

CatalogSchemaSwitch

Default Value	Data Type	Required
0	Integer	No

Description

This property specifies whether the driver treats Spark catalogs as schemas or as catalogs.

- 1: The driver treats Spark catalogs as schemas as a restriction for filtering.
- 0: Spark catalogs are treated as catalogs, and Spark schemas are treated as schemas.

DecimalColumnScale

Default Value	Data Type	Required
10	Integer	No

Description

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

DefaultStringColumnLength

Default Value	Data Type	Required
255	Integer	No

Description

The maximum number of characters that can be contained in STRING columns. The range of `DefaultStringColumnLength` is `0` to `32767`.

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

DelegationUID

Default Value	Data Type	Required
None	String	No

Description

Use this option to delegate all operations against Spark to a user that is different than the authenticated user for the connection.

Note:

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

DnsResolver

Default Value	Data Type	Required
None	String	No

Description

The fully-qualified class path for a class that extends the `DnsResolver` interface provided by the driver, `com.interfaces.networking.CustomDnsResolver`. Using a custom `DnsResolver` enables you to provide your own resolver logic.

DnsResolverArg

Default Value	Data Type	Required
None	String	No

Description

A string argument to pass to the constructor of the class indicated by the `DnsResolver` property.

FastConnection

Default Value	Data Type	Required
`0`	Integer	No

Description

This property specifies whether the driver bypasses the connection testing process. Enabling this option can speed up the connection process, but may result in errors.

- `1`: The driver connects to the data source without first testing the connection.
- `0`: The driver tests the connection before connecting to the data source.

httpPath

Default Value	Data Type	Required
`None`	String	Yes, if `transportMode=http`.

Description

The partial URL corresponding to the Spark server.

The driver forms the HTTP address to connect to by appending the `httpPath` value to the host and port specified in the connection URL. For example, to connect to the HTTP address `http://localhost:10002/cliservice`, you would use the following connection URL:

```
jdbc:spark://localhost:10002;AuthMech=3;transportMode=http;
httpPath=cliservice;UID=jsmith;PWD=simba123;
```

Note:

By default, Spark servers use `cliservice` as the partial URL.

KrbAuthType

Default Value	Data Type	Required
0	Integer	No

Description

This property specifies how the driver obtains the Subject for Kerberos authentication.

- `0`: The driver automatically detects which method to use for obtaining the Subject:
  1. First, the driver tries to obtain the Subject from the current thread's inherited AccessControlContext. If the AccessControlContext contains multiple Subjects, the driver uses the most recent Subject.
  2. If the first method does not work, then the driver checks the `java.security.auth.login.config` system property for a JAAS configuration. If a JAAS configuration is specified, the driver uses that information to create a LoginContext and then uses the Subject associated with it.
  3. If the second method does not work, then the driver checks the KRB5_CONFIG and KRB5CCNAME system environment variables for a Kerberos ticket cache. The driver uses the information from the cache to create a LoginContext and then uses the Subject associated with it.
- `1`: The driver checks the `java.security.auth.login.config` system property for a JAAS configuration. If a JAAS configuration is specified, the driver

uses that information to create a LoginContext and then uses the Subject associated with it.

- `2`: The driver checks the KRB5_CONFIG and KRB5CCNAME system environment variables for a Kerberos ticket cache. The driver uses the information from the cache to create a LoginContext and then uses the Subject associated with it.

KrbHostFQDN

Default Value	Data Type	Required
None	String	Yes, if `AuthMech=1`.

Description

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

KrbRealm

Default Value	Data Type	Required
Depends on your Kerberos configuration	String	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 property.

KrbServiceName

Default Value	Data Type	Required
None	String	Yes, if `AuthMech=1`.

Description

The Kerberos service principal name of the Spark server.

LogLevel

Default Value	Data Type	Required
0	Integer	No

Description

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

Important:

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

The settings for logging apply to every connection that uses the Simba Spark JDBC Driver, so make sure to disable the feature after you are done using it.

Set the property to one of the following numbers:

- 0: Disable all logging.
- 1: Enable logging on the FATAL level, which logs very severe error events that will lead the driver to abort.
- 2: Enable logging on the ERROR level, which logs error events that might still allow the driver to continue running.
- 3: Enable logging on the WARNING level, which logs events that might result in an error if action is not taken.
- 4: Enable logging on the INFO level, which logs general information that describes the progress of the driver.
- 5: Enable logging on the DEBUG level, which logs detailed information that is useful for debugging the driver.
- 6: Enable logging on the TRACE level, which logs all driver activity.

When logging is enabled, the driver produces the following log files in the location specified in the `LogPath` property:

- A `SparkJDBC_driver.log` file that logs driver activity that is not specific to a connection.
- A `SparkJDBC_connection_[Number].log` file for each connection made to the database, where *[Number]* is a number that distinguishes each log file from the others. This file logs driver activity that is specific to the connection.

If the `LogPath` value is invalid, then the driver sends the logged information to the standard output stream (`System.out`).

LogPath

Default Value	Data Type	Required
The current working directory.	String	No

Description

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

Note:

To make sure that the connection URL is compatible with all JDBC applications, it is recommended that you escape the backslashes (`\`) in your file path by typing another backslash.

PreparedMetaLimitZero

Default Value	Data Type	Required
`0`	Integer	No

Description

This property specifies whether the `PreparedStatement.getMetadata()` call will request metadata from the server with `LIMIT 0`.

- `1`: The `PreparedStatement.getMetadata()` call uses `LIMIT 0`.
- `0`: The `PreparedStatement.getMetadata()` call does not use `LIMIT 0`.

PWD

Default Value	Data Type	Required
None	String	Yes, if `AuthMech=3`.

Description

The password corresponding to the user name that you provided using the property UID on page 49.

RowsFetchedPerBlock

Default Value	Data Type	Required
10000	Integer	No

Description

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

Any positive 32-bit integer is a valid value, but testing has shown that performance gains are marginal beyond the default value of 10000 rows.

ServerVersion

Default Value	Data Type	Required
AUTO	String	No

Description

The version number of the data server. This option is used to bypass the driver's server version check. This can speed up the connection process, but may result in errors.

If this option is not set or is set to AUTO, the driver checks the version of the server when a connection is made.

Otherwise, this option should be set to the version number of the server, in the format:

```
[MajorVersion].[MinorVersion].[PatchNumber]
```

For example, ServerVersion=2.4.0 indicates that the driver is connecting to Spark 2.4.0.

! Important:

If this option is set, it must match the version of the server, otherwise errors may occur.

SocketFactory

Default Value	Data Type	Required
None	String	No

Description

The fully-qualified class path for a class that extends `javax.net.SocketFactory` and provides the socket factory implementation. Using a custom SocketFactory enables you to customize the socket that the driver uses.

SocketFactoryArg

Default Value	Data Type	Required
None	String	No

Description

A string argument to pass to the constructor of the class indicated by the `SocketFactory` property.

SocketTimeout

Default Value	Data Type	Required
0	Integer	No

Description

The number of seconds that the TCP socket waits for a response from the server before raising an error on the request.

When this property is set to `0`, the connection does not time out.

SSL

Default Value	Data Type	Required
0	Integer	No

Description

This property specifies whether the driver communicates with the Spark server through an SSL-enabled socket.

- `1`: The driver connects to SSL-enabled sockets.
- `0`: The driver does not connect to SSL-enabled sockets.

Note:

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

SSLKeyStore

Default Value	Data Type	Required
None	String	No

Description

The full path of the Java KeyStore containing the server certificate for one-way SSL authentication.

See also the property SSLKeyStorePwd on page 46.

Note:

The Simba Spark JDBC Driver accepts TrustStores and KeyStores for one-way SSL authentication. See also the property SSLTrustStore on page 46.

SSLKeyStorePwd

Default Value	Data Type	Required
None	Integer	Yes, if you are using a KeyStore for connecting over SSL.

Description

The password for accessing the Java KeyStore that you specified using the property SSLKeyStore on page 45.

SSLTrustStore

Default Value	Data Type	Required
`jssecacerts`, if it exists. If `jssecacerts` does not exist, then `cacerts` is used. The default location of `cacerts` is `jre\lib\security\`.	String	No

Description

The full path of the Java TrustStore containing the server certificate for one-way SSL authentication.

See also the property SSLTrustStorePwd on page 47.

Note:

The Simba Spark JDBC Driver accepts TrustStores and KeyStores for one-way SSL authentication. See also the property SSLKeyStore on page 45.

SSLTrustStorePwd

Default Value	Data Type	Required
None	String	Yes, if using a TrustStore

Description

The password for accessing the Java TrustStore that you specified using the property SSLTrustStore on page 46.

StripCatalogName

Default Value	Data Type	Required
1	Integer	No

Description

This property specifies whether the driver removes catalog names from query statements if translation fails or if the `UseNativeQuery` property is set to `1`.

- `1`: If query translation fails or if the `UseNativeQuery` property is set to `1`, then the driver removes catalog names from the query statement.
- `0`: The driver does not remove catalog names from query statements.

SubjectAlternativeNamesHostNames

Default Value	Data Type	Required
0	Integer	No

Description

This property specifies whether the driver uses the subject alternative names of the SSL certificate instead of the common name when checking if the certificate name matches the host name of the Spark server.

- `0`: The driver checks if the common name of the certificate matches the host name of the server.
- `1`: The driver checks if the subject alternative names of the certificate match the host name of the server.

Note:

This property is applicable only when SSL with one-way authentication is enabled, and the `CAIssuedCertsMismatch` property is also enabled.

TransportMode

Default Value	Data Type	Required
`sasl`	String	No

Description

The transport protocol to use in the Thrift layer.

- `binary`: The driver uses the Binary transport protocol.

  If you use this setting and do not specify the `AuthMech` property, then the driver uses `AuthMech=0` by default. This setting is valid only when the `AuthMech` property is set to `0` or `3`.
- `sasl`: The driver uses the SASL transport protocol.

  If you use this setting but do not specify the `AuthMech` property, then the driver uses `AuthMech=2` by default. This setting is valid only when the `AuthMech` property is set to `1`, `2`, or `3`.
- `http`: The driver uses the HTTP transport protocol.

  If you use this setting but do not specify the `AuthMech` property, then the driver uses `AuthMech=3` by default. This setting is valid only when the `AuthMech` property is set to `3`.

  If you set this property to `http`, then the port number in the connection URL corresponds to the HTTP port rather than the TCP port, and you must specify the `httpPath` property. For more information, see httpPath on page 38.

UID

Default Value	Data Type	Required
`spark`	String	Yes, if `AuthMech=3`.

Description

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

UseNativeQuery

Default Value	Data Type	Required
`0`	Integer	No

Description

This property specifies whether the driver transforms the queries emitted by applications.

- `1`: The driver does not transform the queries emitted by applications, so the native query is used.
- `0`: The driver transforms the queries emitted by applications and converts them into an equivalent form in HiveQL.

Note:

If the application is Spark-aware and already emits HiveQL, then enable this option to avoid the extra overhead of query transformation.

Third-Party Trademarks

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

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.