



Cloudera JDBC Connector for Apache Hive

Important Notice

© 2010-2025 Cloudera, Inc. All rights reserved.

Cloudera, the Cloudera logo, and any other product or service names or slogans contained in this document, except as otherwise disclaimed, are trademarks of Cloudera and its suppliers or licensors, and may not be copied, imitated or used, in whole or in part, without the prior written permission of Cloudera or the applicable trademark holder.

Hadoop and the Hadoop elephant logo are trademarks of the Apache Software Foundation. All other trademarks, registered trademarks, product names and company names or logos mentioned in this document are the property of their respective owners. Reference to any products, services, processes or other information, by trade name, trademark, manufacturer, supplier or otherwise does not constitute or imply endorsement, sponsorship or recommendation thereof by us.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Cloudera.

Cloudera may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Cloudera, the furnishing of this document does not give you any license to these patents, trademarks copyrights, or other intellectual property.

The information in this document is subject to change without notice. Cloudera shall not be liable for any damages resulting from technical errors or omissions which may be present in this document, or from use of this document.

Cloudera, Inc.
1001 Page Mill Road, Building 2
Palo Alto, CA 94304-1008
info@cloudera.com
US: 1-888-789-1488
Intl: 1-650-843-0595
www.cloudera.com

Release Information

Version: 2.6

Date: November 2025

Contents

Contents	3
About the Cloudera JDBC Connector for Apache Hive	7
System Requirements	8
Cloudera JDBC Connector for Apache Hive Files	9
Installing and Using the Cloudera JDBC Connector for Apache Hive	10
Referencing the JDBC Connector Libraries	10
Registering the Connector Class	10
Building the Connection URL	11
Configuring Authentication	13
Using No Authentication	13
Using JSON Web Token (JWT)	13
Using Kerberos	14
Using User Name	15
Using User Name And Password	15
Using a Hadoop Delegation Token	16
Authentication Mechanisms	18
Configuring Kerberos Authentication for Windows	20
Kerberos Encryption Strength and the JCE Policy Files Extension	25
Configuring SSL	27
Configuring Server-Side Properties	29
Configuring Logging	30
Features	32

Contents

SQL Query versus HiveQL Query	32
Data Types	32
Catalog and Schema Support	33
Write-back	33
IHadoopStatement	34
IHadoopConnection	40
Security and Authentication	44
Interfaces and Supported Methods	44
Connector Configuration Options	100
AllowSelfSignedCerts	100
AsyncExecPollInterval	100
AuthMech	101
BinaryColumnLength	101
CAIssuedCertsMismatch	101
CatalogSchemaSwitch	102
DecimalColumnScale	102
DefaultStringColumnLength	102
DelegationToken	102
DelegationUID	103
EnableServerLogs	103
FastConnection	103
httpPath	103
JWTString	104

IgnoreTransactions	104
KrbAuthType	104
KrbHostFQDN	105
KrbRealm	105
KrbServiceName	106
LoginTimeout	106
LogLevel	106
LogPath	107
NonRowcountQueryPrefixes	108
PreparedMetaLimitZero	108
PWD	108
RowsFetchedPerBlock	108
ServerLogsPoolInterval	109
SocketTimeout	109
SSL	109
SSLKeyStore	109
SSLKeyStoreProvider	110
SSLKeyStorePwd	110
SSLKeyStoreType	110
SSLTrustStore	110
SSLTrustStoreProvider	111
SSLTrustStorePwd	111
SSLTrustStoreType	111

Contents

TransportMode	111
UID	112
UseNativeQuery	112
zk	113
Contact Us	114
Third-Party Trademarks	115

About the Cloudera JDBC Connector for Apache Hive

The Cloudera JDBC Connector for Apache Hive is used for direct SQL and HiveQL access to Apache Hadoop / Hive distributions, enabling Business Intelligence (BI), analytics, and reporting on Hadoop / Hive-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 Hive-aware, then the connector is configurable to pass the query through to the database for processing. The connector interrogates Hive 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](#).

The Cloudera JDBC Connector for Apache Hive complies with the JDBC 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 connector, 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>.

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

System Requirements

For details on the specific versions of the data source and Java runtimes supported, please refer to the driver's release notes.

Cloudera JDBC Connector for Apache Hive Files

The Cloudera JDBC Connector for Apache Hive is delivered in the following ZIP archives, where *[Version]* is the version number of the connector:

- `HiveJDBC41_[Version].zip`
- `HiveJDBC42_[Version].zip`

The archive contains the connector 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 connector JAR file in the archive.

Installing and Using the Cloudera JDBC Connector for Apache Hive

To install the Cloudera JDBC Connector for Apache Hive on your machine, extract the files from the appropriate ZIP archive to the directory of your choice.

To access a Hive data store using the Cloudera JDBC Connector for Apache Hive, you need to configure the following:

- The list of connector library files (see [Referencing the JDBC Connector Libraries](#))
- The `Driver` or `DataSource` class (see [Registering the Connector Class](#)).
- The connection URL for the connector (see [Building the Connection URL](#))

Referencing the JDBC Connector Libraries

Before you use the Cloudera JDBC Connector for Apache Hive, the JDBC application or Java code that you are using to connect to your data must be able to access the connector JAR files. In the application or code, specify all the JAR files that you extracted from the ZIP archive.

Using the Connector in a JDBC Application

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

Using the Connector in Java Code

You must include all the connector 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 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/unix/classpath.html>

Registering the Connector Class

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

The following classes are used to connect the Cloudera JDBC Connector for Apache Hive to Hivedata stores:

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

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

- `com.cloudera.hive.jdbc.HS2Driver`
- `com.cloudera.hive.jdbc.HS2DataSource`

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

```
private static Connection connectViaDM() throws Exception
{
    Connection connection = null;
    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;
    DataSource ds = new com.cloudera.hive.jdbc.HS2DataSource();
    ds.setURL(CONNECTION_URL);
    connection = ds.getConnection();
    return connection;
}
```

Building the Connection URL

Use the connection URL to supply connection information to the data source that you are accessing. The following is the format of the connection URL for the Cloudera JDBC Connector for Apache Hive, where *[Subprotocol]* is **hive**, **hive2** if you are connecting to a Hive Server 2 instance, *[Host]* is the DNS or IP address of the Hive server, and *[Port]* is the number of the TCP port that the server uses to listen for client requests:

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



Note:

By default, Hive uses port 10000.

By default, the connector uses the schema named **default** and authenticates the connection using the user name **anonymous**.

You can specify optional settings such as the number of the schema to use or any of the connection properties supported by the connector. For a list of the properties available in the connector, see [Connector Configuration Options](#).

**Note:**

If you specify a property that is not supported by the connector, then the connector attempts to apply the property as a Hive server-side property for the client session. For more information, see [Configuring Server-Side Properties](#).

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

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

For example, to connect to port 11000 on a Hive Server 2 instance 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:hive2://localhost:11000/default2;AuthMech=3;  
UID=cloudera;PWD=cloudera
```

**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, run the `show databases` command at the Hive command prompt.
- If you set the `transportMode` property to `http`, then the port number specified in the connection URL corresponds to the HTTP port rather than the TCP port. By default, Hive servers use 10001 as the HTTP port number.

Configuring Authentication


The Cloudera JDBC Connector for Apache Hive supports the following authentication mechanisms:

- No Authentication
- Kerberos
- JWT
- User Name
- User Name And Password
- Single Sign-On (SSO)
- Hadoop Delegation Token

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

For information about selecting an appropriate authentication mechanism when using the Cloudera JDBC Connector for Apache Hive, see [Configuring Authentication](#).

For information about the properties you can use in the connection URL, see [Connector Configuration Options](#).

 **Note:** In addition to authentication, you can configure the connector to connect over SSL. For more information, see [Configuring SSL](#).

Using No Authentication

You provide this information to the connector in the connection URL. For more information about the syntax of the connection URL, see [Building the Connection URL](#).

To configure a connection without authentication:

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

For example:

```
jdbc:hive2://localhost:10000;AuthMech=0;transportMode=binary;
```

Using JSON Web Token (JWT)

This authentication mechanism requires `JWTString` to be specified in the connection string. If `JWTString` is not specified in the connection string, the connector will attempt to find it in the system environment variables.

You provide this information to the connector in the connection URL. For more information about the syntax of the connection URL, see [Building the Connection URL](#).

Configuring Authentication

To configure JWT authentication:

1. Set the `AuthMech` property to 14.
2. Set the `JWTString` property to the appropriate JWT to access the server.

For example:

```
jdbc:hive://node1.example.com:21050;AuthMech=14;JWTString=s0m3t0ken5tr1ngh3re
```

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](#). For other operating systems, see the MIT Kerberos documentation: <http://web.mit.edu/kerberos/krb5-latest/doc/>.

You provide this information to the connector in the connection URL. For more information about the syntax of the connection URL, see [Building the Connection URL](#).



Note:

- This authentication mechanism is available only for Hive Server 2.
- For this authentication mechanism, SASL and HTTP Thrift transport protocols are supported. If the `transportMode` property is not set, the connector defaults SASL. If the `hive.server2.transport.mode` property has been set to HTTP on the server side, set the `transportMode` property to `http`.

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 Hive server is not the default, then set the `KrbRealm` property to the realm of the Hive server.
3. Set the `KrbHostFQDN` property to the fully qualified domain name of the Hive server host.
4. Optionally, specify how the connector obtains the Kerberos Subject by setting the `KrbAuthType` property as follows:
 - To configure the connector to automatically detect which method to use for obtaining the Subject, set the `KrbAuthType` property to 0. Alternatively, do not set the `KrbAuthType` property.
 - Or, to create a `LoginContext` from a JAAS configuration and then use the Subject associated with it, set the `KrbAuthType` property to 1.
 - Or, to create a `LoginContext` from a Kerberos ticket cache and then use the Subject associated with it, set the `KrbAuthType` property to 2.

For more detailed information about how the connector obtains Kerberos Subjects based on these settings, see [KrbAuthType](#).

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

```
jdbc:hive2://node1.example.com:10000;AuthMech=1;  
KrbRealm=EXAMPLE.COM;KrbHostFQDN=hs2node1.example.com;  
KrbServiceName=hive;KrbAuthType=2
```

In this example, Kerberos is enabled for JDBC connections, the Kerberos service principal name is `hive/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 connector in the connection URL. For more information about the syntax of the connection URL, see [Building the Connection URL](#).

**Note:**

This authentication mechanism is available only for Hive Server 2. Most default configurations of Hive Server 2 require User Name authentication.

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 Hive server.

For example:

```
jdbc:hive2://node1.example.com:10000;AuthMech=2;  
transportMode=sasl;UID=hs2
```

Using User Name And Password

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

You provide this information to the connector in the connection URL. For more information about the syntax of the connection URL, see [Building the Connection URL](#).

**Note:**

This authentication mechanism is available only for Hive Server 2.

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.

Configuring Authentication

3. If you set the `transportMode` property to `http`, then set the `httpPath` property to the partial URL corresponding to the Hive server. Otherwise, do not set the `httpPath` property.
4. Set the `UID` property to an appropriate user name for accessing the Hive 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 Hive server with LDAP authentication enabled:

```
jdbc:hive2://node1.example.com:10001;AuthMech=3;  
transportMode=http;httpPath=cliservice;UID=hs2;PWD=cloudera;
```

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

Using a Hadoop Delegation Token

This authentication mechanism requires a Hadoop delegation token. This token must be provided to the connector in the form of a Base64 URL-safe encoded string. It can be obtained from the connector using the `getDelegationToken()` function, or by utilizing the Hadoop distribution `.jar` files. For a code sample that demonstrates how to retrieve the token using the `getDelegationToken()` function, see [Code Samples: Retrieving a Hadoop Delegation Token](#).

You provide this information to the connector in the connection URL. For more information about the syntax of the connection URL, see [Building the Connection URL](#).



Note:

- This authentication mechanism is available only for Hive Server 2.
- This authentication mechanism requires that Kerberos be configured on the server.

To configure Hadoop delegation token authentication:

1. Make sure Kerberos is configured on the server.
2. Set the `AuthMech` property to 6.
3. Set the `delegationToken` property to an appropriately encoded Hadoop delegation token.

For example:

```
jdbc:hive2://node1.example.com:10000  
;AuthMech=6;delegationToken=kP9PcyQ7prK2LwUMZMpFQ4R+5VE
```

Code Samples: Retrieving a Hadoop Delegation Token

If you are using a Hadoop delegation token for authentication, the token must be provided to the connector in the form of a Base64 URL-safe encoded string. This token can be obtained from the connector using the `getDelegationToken()` function, or by utilizing the Hadoop distribution `.jar` files.

The code samples below demonstrate the use of the `getDelegationToken()` function. For more information about this function, see [IHadoopConnection](#).

The sample below shows how to obtain the token string with the connector using a Kerberos connection:

```
import com.cloudera.hiveserver2.hivecommon.core.IHadoopConnection;

public class TestDriverGetDelegationTokenClass
{
    public static void main(String[] args) throws SQLException
    {
        // Create the connection object with Kerberos authentication.
        Connection kerbConnection = DriverManager.getConnection(
            "jdbc:hive2://localhost:10000;AuthMech=1;KrbRealm=YourR
            ealm;KrbHostFQDN=sample.com;KrbServiceName=hive;");

        // Unwrap the java.sql.Connection object to an implementation of
        // IHadoopConnection so the
        // methods for delegation token can be called.
        String delegationToken = kerbConnection.unwrap
            (IHadoopConnection.class).getDelegationToken("owner_name", "renewer_
            name");

        // The token can then be used to connect with the connector.
        String tokenConnectionString =
            "jdbc:hive2://localhost:10000;AuthMech=6;DelegationToken=" +
            delegationToken;

        Connection tokenConnection = DriverManager.getConnection
            (tokenConnectionString);
    }
}
```

The sample below demonstrates how to obtain the encoded string form of the token if the delegation is saved to the UserGroupInformation. This sample requires the `hadoop-shims-common-[hadoop version].jar`, `hadoop-common-[hadoop version].jar`, and all their dependencies.

```
import org.apache.hadoop.hive.shims.Utils;
import org.apache.hive.service.auth.HiveAuthFactory;

public class TestHadoopDelegationTokenClass
{
    public static void main(String[] args) throws SQLException
```

```
{  
  
    // Obtain the delegationToken stored in the current UserGroupInformation.  
    String delegationToken = Utils.getTokenStrForm(HiveAuthFactory.HS2_  
    CLIENT_TOKEN);  
  
    // The token can then be used to connect with the connector.  
    String tokenConnectionString =  
    "jdbc:hive2://localhost:10000;AuthMech=6;DelegationToken=" +  
    delegationToken;  
    Connection tokenConnection = DriverManager.getConnection  
    (tokenConnectionString);  
}  
}
```

Authentication Mechanisms

To connect to a Hive server, you must configure the Cloudera JDBC Connector for Apache Hive 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 Hive server requires, check the server configuration and then refer to the corresponding section below.

Hive Server 2

Hive Server 2 supports the following authentication mechanisms:

- No Authentication (see [Using No Authentication](#))
- JWT (see [Using JSON Web Token \(JWT\)](#))
- Kerberos (see [Using Kerberos](#))
- User Name (see [Using User Name](#))
- User Name And Password (see [Using User Name And Password](#))
- Hadoop Delegation Token (see [Using a Hadoop Delegation Token](#))

Most default configurations of Hive Server 2 require User Name authentication. If you are unable to connect to your Hive server using User Name authentication, then verify the authentication mechanism configured for your Hive 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 Hive 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 Hive 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 connector based on the settings in the `hive-site.xml` file.

hive.server2.authentication	hive.server2.enable.doAs	Connector 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 / Hive 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 Hive Server 2 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 Hive Server 2 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 connector from the connector configuration to an existing operating system user on the host running Hive Server 2. 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 connector configuration is ignored.

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

Using User Name And Password

When connecting to a Hive Server 2 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 connector 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 Hive Server

You provide this information to the connector in the connection URL. For more information about the syntax of the connection URL, see [Building the Connection URL](#).

To authenticate to the Hive server:

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

Configuring Authentication

- AuthMech
- KrbHostFQDN
- KrbRealm
- KrbServiceName


For detailed information about these properties, see [Connector Configuration Options](#)

Using the Connector to Get Tickets

Deleting the KRB5CCNAME Environment Variable

To enable the connector 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="cloudera@CLLOUDERA"
```

```
doNotPrompt=true;
};
```

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

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



Note:

JAAS configuration is disabled by default. To enable JAAS configuration, please set the `JDBC_ENABLE_JAAS` environment variable to 1.

Authenticating to the Hive Server

You provide this information to the connector in the connection URL. For more information about the syntax of the connection URL, see [Building the Connection URL](#).

To authenticate to the Hive server:

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

For detailed information about these properties, see [Connector Configuration Options](#).

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
```

Configuring Authentication

```
static String ConnectionURL = "jdbc:hive2://192.168.1.1:10000";  
/**  
 * Logic executed in this method will have access to the  
 * Subject that is used to "doAs". The connector 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;

```

Kerberos Encryption Strength and the JCE Policy Files Extension

If the encryption being used in your Kerberos environment is too strong, you might encounter the error message "Unable to connect to server: GSS initiate failed" when trying to use the connector to connect to a Kerberos-enabled cluster. Typically, Java vendors only allow encryption strength up to 128 bits by default. If you are using greater encryption strength in your environment (for example, 256-bit encryption), then you might encounter this error.

Diagnosing the Issue

If you encounter the error message "Unable to connect to server: GSS initiate failed", confirm that it is occurring due to encryption strength by enabling Kerberos layer logging in the JVM and then checking if the log output contains the error message "KrbException: Illegal key size".

To enable Kerberos layer logging in a Sun JVM:

- Choose one:
 - In the Java command you use to start the application, pass in the following argument:


```
-Dsun.security.krb5.debug=true
```
 - Or, add the following code to the source code of your application:


```
System.setProperty("sun.security.krb5.debug", "true")
```

To enable Kerberos layer logging in an IBM JVM:

- Choose one:
 - In the Java command you use to start the application, pass in the following arguments:


```
-Dcom.ibm.security.krb5.Krb5Debug=all
-Dcom.ibm.security.jgss.debug=all
```
 - Or, add the following code to the source code of your application:


```
System.setProperty("com.ibm.security.krb5.Krb5Debug", "all");
System.setProperty("com.ibm.security.jgss.debug", "all");
```

Resolving the Issue

After you confirm that the error is occurring due to encryption strength, you can resolve the issue by downloading and installing the *Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files* extension from your Java vendor. Refer to the instructions from the vendor to install the files to the correct location.



Important:

Consult your company's policy to make sure that you are allowed to enable encryption strengths in your environment that are greater than what the JVM allows by default.

If the issue is not resolved after you install the JCE policy files extension, then restart your machine and try your connection again. If the issue persists even after you restart your machine, then verify which directories the JVM is searching to find the JCE policy files extension. To print out the search paths that your JVM currently uses to find the JCE policy files extension, modify your Java source code to print the return value of the following call:

```
System.getProperty("java.ext.dirs")
```

Configuring SSL

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

If you are connecting to a Hive server that has Secure Sockets Layer (SSL) enabled, you can configure the connector to connect to an SSL-enabled socket. When connecting to a server over SSL, the connector 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 connector to access a specific TrustStore or KeyStore that contains the appropriate certificate. If you do not specify a TrustStore or KeyStore, then the connector uses the default Java TrustStore named `jssecacerts`. If `jssecacerts` is not available, then the connector uses `cacerts` instead.

You provide this information to the connector in the connection URL. For more information about the syntax of the connection URL, see [Building the Connection URL](#).

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 connector to use it:
 - Create a TrustStore containing your signed, trusted server certificate.
 - Set the `SSLTrustStore` property to the full path of the TrustStore.
 - Set the `SSLTrustStorePwd` property to the password for accessing the TrustStore.
 - If the TrustStore is not a JKS TrustStore, set the `SSLTrustStoreType` property to the correct type. The supported types are:
 - `SSLTrustStoreType=BCFKS` (BouncyCastle FIPS Keystore)
 - `SSLTrustStoreType=PKCS12` (Public Key Cryptography Standards #12)



Note:

`SSLTrustStoreType=PKCS11` (Public Key Cryptography Standards #11) TrustStore type is not supported.

- To specify a Java Security API provider, set the `SSLTrustStoreProvider` property to the name of the provider.
- Or, create a KeyStore and configure the connector to use it:
 - Create a KeyStore containing your signed, trusted server certificate.
 - Set the `SSLKeyStore` property to the full path of the KeyStore.

Configuring SSL

- Set the `SSLKeyStorePwd` property to the password for accessing the KeyStore.
 - If the KeyStore is not a JKS KeyStore, set the `SSLKeyStoreType` property to the correct type.
 - To specify a Java Security API provider, set the `SSLKeyStoreProvider` property to the name of the provider.
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 connector 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 Hive server, set the `CAIssuedCertNamesMismatch` property to 1.

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

```
jdbc:hive2://localhost:10000;AuthMech=3;SSL=1;  
SSLKeyStore=C:\\Users\\bsmith\\Desktop\\keystore.jks;SSLKeyStorePwd=cloudera  
SSL123;UID=hs2;PWD=cloudera123
```



Note:

For more information about the connection properties used in SSL connections, see [Connector Configuration Options](#).

Configuring Server-Side Properties

You can use the connector to apply configuration properties to the Hive server by setting the properties in the connection URL.

For example, to set the `mapreduce.job.queueName` property to `myQueue`, you would use a connection URL such as the following:

```
jdbc:hive://localhost:18000/default2;AuthMech=3;  
UID=cloudera;PWD=cloudera;mapreduce.job.queueName=myQueue
```



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

Configuring Logging

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

**Important:**

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

The settings for logging apply to every connection that uses the Cloudera JDBC Connector for Apache Hive, 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 Cloudera JDBC Connector for Apache Hive, in order from least verbose to most verbose.

LogLevel Value	Description
0	Disable all logging.
1	Log severe error events that lead the connector to abort.
2	Log error events that might allow the connector 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 connector.
5	Log detailed information that is useful for debugging the connector.
6	Log all connector 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:hive://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 Cloudera JDBC Connector for Apache Hive produces the following log files in the location specified in the `LogPath` property:

- A `HiveJDBC_driver.log` file that logs connector activity that is not specific to a connection.

- A `HiveJDBC_connection_[Number].log` file for each connection made to the database, where *[Number]* is a number that identifies each log file. This file logs connector activity that is specific to the connection.

If the `LogPath` value is invalid, then the connector 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 Cloudera JDBC Connector for Apache Hive:

- [SQL Query versus HiveQL Query](#)
- [Data Types](#)
- [Catalog and Schema Support](#)
- [Write-back](#)
- [IHadoopStatement](#)
- [IHadoopConnection](#)
- [Security and Authentication](#)
- [Interfaces and Supported Methods](#)

SQL Query versus HiveQL Query

The native query language supported by Hive 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 Cloudera JDBC Connector for Apache Hive supports many common data formats, converting between Hive, SQL, and Java data types.

The following table lists the supported data type mappings.

Hive Type	SQL Type	Java Type
BIGINT	BIGINT	java.math.BigInteger
BINARY	VARBINARY	byte[]
BOOLEAN	BOOLEAN	Boolean
CHAR (Available only in Hive 0.13.0 or later)	CHAR	String
DATE	DATE	java.sql.Date
DECIMAL (In Hive 0.13 and later, you can specify scale and precision when creating tables using the DECIMAL data type.)	DECIMAL	java.math.BigDecimal

Hive Type	SQL Type	Java Type
DOUBLE	DOUBLE	Double
FLOAT	REAL	Float
INT	INTEGER	Long
SMALLINT	SMALLINT	Integer
TIMESTAMP	TIMESTAMP	java.sql.Timestamp
TINYINT	TINYINT	Short
VARCHAR (Available only in Hive 0.12.0 or later)	VARCHAR	String

The aggregate types (ARRAY, MAP, STRUCT, and UNIONTYPE) are not yet supported. Columns of aggregate types are treated as VARCHAR columns in SQL and STRING columns in Java.

Catalog and Schema Support

The Cloudera JDBC Connector for Apache Hive supports both catalogs and schemas to make it easy for the connector to work with various JDBC applications. Since Hive only organizes tables into schemas/databases, the connector provides a synthetic catalog named HIVE under which all of the schemas/databases are organized. The connector also maps the JDBC schema to the Hive schema/database.



Note:

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

Write-back

The Cloudera JDBC Connector for Apache Hive supports translation for the following syntax when connecting to a Hive Server 2 instance that is running Hive 0.14 or later:

- INSERT
- UPDATE
- DELETE
- CREATE
- DROP

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.

IHadoopStatement

IHadoopStatement is an interface implemented by the connector's statement class. It provides access to methods that allow for asynchronous execution of queries and the retrieval of the Yarn ATS GUID associated with the execution.

The IHadoopStatement interface is defined by the `IHadoopStatement.java` file. This file should look like the following example:

```
//
=====

/// @file IHadoopStatement.java /// /// Exposed interface for asynchronous query execution. /// ///
Copyright (C) 2017 Simba Technologies Incorporated.

//
=====

package com.cloudera.hiveserver2.hivecommon.core;
import java.sql.ResultSet; import java.sql.SQLException;
import java.sql.Statement;
/**
 * An interface that extends the standard SQL Statement Interface but allows for asynchronous
 * query execution.
 * The polling for query execution will occur when {@link ResultSet#next()} or
 * {@link ResultSet#getMetaData()} is called.
 */ public interface IHadoopStatement extends Statement
{
    /**
     * Executes the given SQL statement asynchronously.
     * <p> * Sends the query off to the server but does not wait for query
     execution to complete.
     * A ResultSet with empty columns is returned.
     * The polling for completion of query execution is done when {@link
     ResultSet#next()} or
     * {@link ResultSet#getMetaData()} is called.
     * </p>
     *
     * @param sql
```

An SQL statement to be sent to the database, typically a

* static SQL SELECT statement.

*

* @return A ResultSet object that DOES NOT contain the data produced by the given query; never null.

*

* @throws SQLException If a database access error occurs, or the given SQL

* statement produces anything other than a single

* `ResultSet` object.

*/

```
public ResultSet executeAsync(String sql) throws SQLException;
```

```
/**
```

* Returns the Yarn ATS guid.

*

* @return String The yarn ATS guid from the operation if execution has started,

* else null.

*/

```
public String getYarnATSGuid(); }
```

The following methods are available for use in `IHadoopStatement`:

■ `executeAsync(String sql)`

The connector sends a request to the server for statement execution and returns immediately after receiving a response from the server for the execute request without waiting for the server to complete the execution.

The connector does not wait for the server to complete query execution unless `getMetaData()` or `next()` APIs are called.

Note that this feature does not work with prepared statements.

For example:

```
import com.cloudera.hiveserver2.hivecommon.core.IHadoopStatement;
```

```
public class TestExecuteAsyncClass
```

```
{
```

```
    public static void main(String[] args) throws SQLException
```

Features

```
{  
  
    // Create the connection object.  
    Connection connection = DriverManager.getConnection  
    ("jdbc:hive2://localhost:10000");  
  
    // Create the statement object.  
    Statement statement = connection.createStatement();  
  
    // Unwrap the java.sql.Statement object to an implementation of  
    IHadoopStatement so the  
    // execution can be done asynchronously.  
    //  
    // The connector will return from this call as soon as it gets a  
    // response from the  
    // server for the execute request without waiting for server to  
    // complete query execution.  
    ResultSet resultSet =  
        statement.unwrap(  
            IHadoopStatement.class).execute  
            Async(  
                "select * from example_table");  
  
    // Calling getMetaData() on the ResultSet here will cause the connector to  
    // wait for the server  
    // to complete query execution before proceeding with the rest of the  
    // operation.  
    ResultSetMetaData rsMetadata = resultSet.getMetaData();  
  
    // Excluding code for work on the result set metadata...  
  
    // Calling getMetaData() on the ResultSet here, and if getMetaData() was  
    // not call prior to  
    // this, will cause the connector to wait for the server to complete query  
    // execution before
```

```

        // proceeding with the rest of the operation.
        resultSet.next();

        // Excluding code for work on the result set ...
    }
}

```

■ `getYarnATSGuid()`

Returns the Yarn ATS GUID associated with the current execution. Returns null if the Yarn ATS GUID is not available.

For example:

```

public class TestYarnGUIDClass
{
    public static void main(String[] args) throws SQLException
    {
        // Create the connection object.
        Connection connection = DriverManager.getConnection
            ("jdbc:hive2://localhost:10000");

        // Create the statement object.
        Statement statement = connection.createStatement();

        // Execute a query.
        ResultSet resultSet = statement.executeQuery("select * from example_
            table");

        // Unwrap the java.sql.Statement object to an implementation
        // of IHadoopStatement to access the
        // getYarnATSGuid() API call.
        String guid = statement.unwrap(
            IHadoopStatement.class).getYarnATSGuid();
    }
}

```

The connector can retrieve the query logs from the server when executing a SQL statement using the following functions:

Features

`getQueryLog(boolean incremental,int fetchSize)`

The connector gets the execution logs of the given SQL statement.

`hasMoreLogs () ;`

The connector checks whether the query execution produces more logs to be fetched.

```
/**
 * Get the execution logs of the given SQL statement.
 *
 * @param incremental True to indicate to get logs incrementally,
 * otherwise false to indicate to get logs from the beginning,
 * @param fetchSize The number of lines to fetch
 *
 * @return A list of logs. It can be empty if there are no new logs to be retrieved at that time.
 * @throws RuntimeException if an error occurs fetching logs from the server
 */
public List<String> getQueryLog(boolean incremental, int fetchSize) throws
RuntimeException;

/**
 * Check whether query execution produces more logs to be fetched.
 *
 * @return true if the query execution produces more logs, false otherwise
 */
public boolean hasMoreLogs();
```

For example:

```
public class TestGetQueryLog
{
    public static void main(String[] args) throws SQLException
    {
```

```

// Create the connection object.
Connection connection = DriverManager.getConnection
("jdbc:hive2://localhost:10000");

// Create statement object.
Statement statement = connection.createStatement();

try
{
    // Create and start the class that gets the query logs in the
    background
    // while the query is running.
    HiveLogRunnable runnable = new HiveLogRunnable
(statement);
    Thread t1 = new Thread(runnable);
    t1.setDaemon(true);
    t1.start();

    // Execute the query.
    ResultSet resultSet = statement.executeQuery("select * from
example_table");
    while (resultSet.next()) {}
}

finally
{
    // Close the statement.
    if (statement != null)
    {
        statement.close();
    }
}
}

```

Features

```
}

class HiveLogRunnable implements Runnable {

    private Statement stmt = null;

    public HiveLogRunnable(Statement stmt) {

        this.stmt = stmt;

    }

    @Override
    public void run() {

        try {

            // Print the query logs to the console while the
            // query has logs.

            while (stmt.unwrap
                (IHadoopStatement.class).hasMoreLogs()) {

                List<String> logLists = stmt.unwrap
                    (IHadoopStatement.class).getQueryLog(true,
                    0);

                for (String string : logLists) {

                    System.out.println(string);

                }

                Thread.sleep(0);

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}
```

IHadoopConnection

IHadoopConnection is an interface implemented by the connector's statement class. It provides access to methods that allow for the retrieval, deletion, and renewal of delegation tokens.

The IHadoopStatement interface is defined by the `IHadoopStatement.java` file. This file should look like the following example:


```
//
=====
=====

/// @file IHadoopConnection.java
///
/// Exposed interface for the retrieval of delegation tokens.
///
/// Copyright (C) 2017 Simba Technologies Incorporated.
//
=====
=====

package com.cloudera.hiveserver2.hivecommon.core;
import java.sql.Connection;
import java.sql.SQLException;
/**
 * An interface that extends the standard SQL Connection Interface but allows for the
 * retrieval/renewal/cancellation of delegation tokens.
 */
public interface IHadoopConnection extends Connection
{
    /**
     * Sends a cancel delegation token request to the server.
     *
     * @param tokenString The token to cancel.
     * @throws SQLException If an error occurs while sending the request.
     */
    public void cancelDelegationToken(String tokenString) throws
        SQLException;

    /**
     * Sends a get delegation token request to the server and returns the token
     * as an
     * encoded string.
     *
     */
}
```

Features

```
* @param owner The owner of the token.
* @param renewer The renewer of the token.
*
* @return The token as an encoded string.
* @throws SQLException If an error occurs while getting the token.
*/

public String getDelegationToken(String owner, String renewer) throws
SQLException;

/**
* Sends a renew delegation token request to the sever.
*
* @param tokenString The token to renew.
* @throws SQLException If an error occurs while sending the request.
*/

public void renewDelegationToken(String tokenString) throws
SQLException;

}
```

The following methods are available for use in `IHadoopConnection`:

- `getDelegationToken(String owner, String renewer)`

The connector sends a request to the server to obtain a delegation token with the given owner and renewer.

The method should be called on a Kerberos-authenticated connection.

- `cancelDelegationToken()`

The connector sends a request to the server to cancel the provided delegation token.

- `renewDelegationToken()`

The connector sends a request to the server to renew the provided delegation token.

The following is a basic code sample that demonstrates how to use the above functions:

```
public class TestDelegationTokenClass
{
    public static void main(String[] args) throws SQLException
    {
        // Create the connection object with Kerberos authentication.
```

```

Connection kerbConnection = DriverManager.getConnection(
    "jdbc:hive2://localhost:10000;AuthMech=1;KrbRealm=YourR
    ealm;KrbHostFQDN=sample.com;KrbServiceName=hive;");

// Unwrap the java.sql.Connection object to an implementation
// of IHadoopConnection so the methods for delegation token
// can be called.
String delegationToken = kerbConnection.unwrap
(IHadoopConnection.class).getDelegationToken("owner_name", "renewer_
name");

// The token can then be used to connect with the connector.
String tokenConnectionString =
"jdbc:hive2://localhost:10000;AuthMech=6;DelegationToken=" +
delegationToken;
Connection tokenConnection = DriverManager.getConnection
(tokenConnectionString);

// Excluding code for work with the tokenConnection ...

// The original token (delegationToken) can be cancelled or renewed by
unwrapping the java.sql.Connection object again to
// an implementation of IHadoopConnection.

// Renewing the token:
kerbConnection.unwrap(IHadoopConnection.class).renewDelegationToken
(delegationToken);

// Cancelling the token:
kerbConnection.unwrap
(IHadoopConnection.class).cancelDelegationToken(delegationToken);
    }
}

```

Security and Authentication

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

**Note:**

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

The connector provides mechanisms that allow you to authenticate your connection using the Kerberos protocol, your Hive user name only, or your Hive user name and password. You must use the authentication mechanism that matches the security requirements of the Hive server. For information about determining the appropriate authentication mechanism to use based on the Hive server configuration, see [Authentication Mechanisms](#). For detailed connector configuration instructions, see [Configuring Authentication](#).

Additionally, the connector supports SSL connections with one-way authentication. If the server has an SSL-enabled socket, then you can configure the connector 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](#).

The SSL version that the connector 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 connector and the server, which is determined at connection time.

Interfaces and Supported Methods

The Cloudera JDBC Connector for Apache Hive implements the following JDBC interfaces:

- [CallableStatement](#)
- [Connection](#)
- [DatabaseMetaData](#)
- [DataSource](#)
- [Driver](#)
- [ParameterMetaData](#)
- [PooledConnection](#)
- [PreparedStatement](#)
- [ResultSet](#)
- [ResultSetMetaData](#)
- [Statement](#)

However, the connector does not support every method from these interfaces. For information about whether a specific method is supported by the connector and which version of the JDBC API is the earliest version that supports the method, refer to the following sections.

The connector does not support the following JDBC features:

- Array
- Blob
- Clob
- Ref
- Savepoint
- SQLData
- SQLInput
- SQLOutput
- Struct

CallableStatement

The `CallableStatement` interface extends the `PreparedStatement` interface.

The following table lists the methods that belong to the `CallableStatement` interface, and describes whether each method is supported by the Cloudera JDBC Connector for Apache Hive and which version of the JDBC API is the earliest version that supports the method.

For detailed information about each method in the `CallableStatement` interface, see the Java API documentation: <http://docs.oracle.com/javase/1.5.0/docs/api/java/sql/CallableStatement.html>.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
Array <code>getArray(int i)</code>	3.0	No	
Array <code>getArray(String parameterName)</code>	3.0	No	
BigDecimal <code>getBigDecimal(int parameterIndex)</code>	3.0	Yes	
BigDecimal <code>getBigDecimal(int parameterIndex, int scale)</code>	3.0	Yes	Deprecated.
BigDecimal <code>getBigDecimal(String parameterName)</code>	3.0	Yes	
Blob <code>getBlob(int i)</code>	3.0	No	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
Blob getBlob(String parameterName)	3.0	No	
boolean getBoolean(int parameterIndex)	3.0	Yes	
boolean getBoolean(String parameterName)	3.0	Yes	
byte getByte(int parameterIndex)	3.0	Yes	
byte getByte(String parameterName)	3.0	Yes	
byte[] getBytes(int parameterIndex)	3.0	Yes	
byte[] getBytes(String parameterName)	3.0	Yes	
Clob getClob(int i)	3.0	No	
Clob getClob(String parameterName)	3.0	No	
Date getDate(int parameterIndex)	3.0	Yes	
Date getDate(int parameterIndex, Calendar cal)	3.0	Yes	
Date getDate(String parameterName)	3.0	Yes	
Date getDate(String parameterName, Calendar cal)	3.0	Yes	
double getDouble(int parameterIndex)	3.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
double getDouble(String parameterName)	3.0	Yes	
float getFloat(int parameterIndex)	3.0	Yes	
float getFloat(String parameterName)	3.0	Yes	
int getInt(int parameterIndex)	3.0	Yes	
int getInt(String parameterName)	3.0	Yes	
long getLong(int parameterIndex)	3.0	Yes	
long getLong(String parameterName)	3.0	Yes	
Reader getNCharacterStream(int parameterIndex)	4.0	No	
Reader getNCharacterStream(String parameterName)	4.0	No	
NClob getNClob(int parameterIndex)	4.0	No	
NClob getNClob(String parameterName)	4.0	No	
String getNString(int parameterIndex)	4.0	No	
String getNString(String parameterName)	4.0	No	
Object getObject(int parameterIndex)	3.0	Yes	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code><T> T getObject(int parameterIndex, Class<T> type)</code>	4.1	No	
<code>Object getObject(int i, Map<String,Class<?>> map)</code>	3.0	No	
<code>Object getObject(String parameterName)</code>	3.0	Yes	
<code><T> T getObject(String parameterName, Class<T> type)</code>	4.1	No	
<code>Object getObject(String parameterName, Map<String,Class<?>> map)</code>	3.0	Yes	
<code>Ref getRef(int i)</code>	3.0	No	
<code>Ref getRef(String parameterName)</code>	3.0	No	
<code>RowId getRowId(int parameterIndex)</code>	4.0	No	
<code>RowId getRowId(String parameterName)</code>	4.0	No	
<code>short getShort(int parameterIndex)</code>	3.0	Yes	
<code>short getShort(String parameterName)</code>	3.0	Yes	
<code>SQLXML getSQLXML(int parameterIndex)</code>	4.0	No	
<code>SQLXML getSQLXML(String parameterName)</code>	4.0	No	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
String getString(int parameterIndex)	3.0	Yes	
String getString(String parameterName)	3.0	Yes	
Time getTime(int parameterIndex)	3.0	Yes	
Time getTime(int parameterIndex, Calendar cal)	3.0	Yes	
Time getTime(String parameterName)	3.0	Yes	
Time getTime(String parameterName, Calendar cal)	3.0	Yes	
Timestamp getTimestamp(int parameterIndex)	3.0	Yes	
Timestamp getTimestamp(int parameterIndex, Calendar cal)	3.0	Yes	
Timestamp getTimestamp (String parameterName)	3.0	Yes	
Timestamp getTimestamp (String parameterName, Calendar cal)	3.0	Yes	
URL getURL(int parameterIndex)	3.0	No	
URL getURL(String parameterName)	3.0	No	
void registerOutParameter(int parameterIndex, int sqlType)	3.0	Yes	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void registerOutParameter(int parameterIndex, int sqlType, int scale)</code>	3.0	Yes	
<code>void registerOutParameter(int paramIndex, int sqlType, String typeName)</code>	3.0	Yes	
<code>void registerOutParameter(String parameterName, int sqlType)</code>	3.0	Yes	
<code>void registerOutParameter(String parameterName, int sqlType, int scale)</code>	3.0	Yes	
<code>void registerOutParameter(String parameterName, int sqlType, String typeName)</code>	3.0	Yes	
<code>void setAsciiStream(String parameterName, InputStream x)</code>	4.0	Yes	
<code>void setAsciiStream(String parameterName, InputStream x, int length)</code>	3.0	Yes	
<code>void setAsciiStream(String parameterName, InputStream x, long length)</code>	4.0	Yes	
<code>void setBigDecimal(String parameterName, BigDecimal x)</code>	3.0	Yes	
<code>void setBinaryStream(String parameterName, InputStream x)</code>	4.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
setBinaryStream(String parameterName, InputStream x, int length)	3.0	Yes	
void setBinaryStream(String parameterName, InputStream x, long length)	4.0	Yes	
void setBlob(String parameterName, Blob x)	4.0	Yes	
void setBlob(String parameterName, InputStream inputStream)	4.0	Yes	
void setBlob(String parameterName, InputStream inputStream, long length)	4.0	Yes	
void setBoolean(String parameterName, boolean x)	3.0	Yes	
void setByte(String parameterName, byte x)	3.0	Yes	
void setBytes(String parameterName, byte[] x)	3.0	Yes	
void setCharacterStream(String parameterName, Reader reader)	4.0	Yes	
void setCharacterStream(String parameterName, Reader reader, int length)	3.0	Yes	
void setCharacterStream(String parameterName, Reader reader, long length)	4.0	Yes	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void setClob(String parameterName, Clob x)</code>	4.0	Yes	
<code>void setClob(String parameterName, Reader reader)</code>	4.0	Yes	
<code>void setClob(String parameterName, Reader reader, long length)</code>	4.0	Yes	
<code>void setDate(String parameterName, Date x)</code>	3.0	Yes	
<code>void setDate(String parameterName, Date x, Calendar cal)</code>	3.0	Yes	
<code>void setDouble(String parameterName, double x)</code>	3.0	Yes	
<code>void setFloat(String parameterName, float x)</code>	3.0	Yes	
<code>void setInt(String parameterName, int x)</code>	3.0	Yes	
<code>void setLong(String parameterName, long x)</code>	3.0	Yes	
<code>void setNCharacterStream(String parameterName, Reader value)</code>	4.0	Yes	
<code>void setNCharacterStream(String parameterName, Reader value, long length)</code>	4.0	Yes	
<code>void setNClob(String parameterName, NClob value)</code>	4.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void setNClob(String parameterName, Reader reader)</code>	4.0	Yes	
<code>void setNClob(String parameterName, Reader reader, long length)</code>	4.0	Yes	
<code>void setNString(String parameterName, String value)</code>	4.0	Yes	
<code>void setNull(String parameterName, int sqlType)</code>	3.0	Yes	
<code>void setNull(String parameterName, int sqlType, String typeName)</code>	3.0	Yes	
<code>void setObject(String parameterName, Object x)</code>	3.0	Yes	
<code>void setObject(String parameterName, Object x, int targetSqlType)</code>	3.0	Yes	
<code>void setObject(String parameterName, Object x, int targetSqlType, int scale)</code>	3.0	Yes	
<code>void setRowId(String parameterName, RowId x)</code>	4.0	Yes	
<code>void setShort(String parameterName, short x)</code>	3.0	Yes	
<code>void setSQLXML(String parameterName, SQLXML xmlObject)</code>	4.0	Yes	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void setString(String parameterName, String x)</code>	3.0	Yes	
<code>void setTime(String parameterName, Time x)</code>	3.0	Yes	
<code>void setTime(String parameterName, Time x, Calendar cal)</code>	3.0	Yes	
<code>void setTimestamp(String parameterName, Timestamp x)</code>	3.0	Yes	
<code>void setTimestamp(String parameterName, Timestamp x, Calendar cal)</code>	3.0	Yes	
<code>void setURL(String parameterName, URL val)</code>	3.0	Yes	
<code>boolean wasNull()</code>	3.0	Yes	
<code>boolean isWrapperFor(Class<?> iface)</code>	4.0	Yes	
<code><T> T unwrap(Class<T> iface)</code>	4.0	Yes	

Connection

The following table lists the methods that belong to the `Connection` interface, and describes whether each method is supported by the Cloudera JDBC Connector for Apache Hive and which version of the JDBC API is the earliest version that supports the method.

For detailed information about each method in the `Connection` interface, see the Java API documentation: <http://docs.oracle.com/javase/1.5.0/docs/api/java/sql/Connection.html>.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void clearWarnings()</code>	3.0	Yes	
<code>void close()</code>	3.0	Yes	
<code>void commit()</code>	3.0	Yes	Auto-commit cannot be set to <code>false</code> because it is hard-coded to <code>true</code> .
Array <code>createArrayOf(String typeName, Object[] elements)</code>	4.0	No	
Blob <code>createBlob()</code>	4.0	No	
Clob <code>createClob()</code>	4.0	No	
NClob <code>createNClob()</code>	4.0	No	
SQLXML <code>createSQLXML()</code>	4.0	No	
Statement <code>createStatement()</code>	3.0	Yes	
Statement <code>createStatement(int resultSetType, int resultSetConcurrency)</code>	3.0	No	
Statement <code>createStatement(int resultSetType, int resultSetConcurrency, int resultSetHoldability)</code>	3.0	No	
Struct <code>createStruct(String typeName, Object[] attributes)</code>	4.0	No	
<code>boolean getAutoCommit()</code>	3.0	Yes	Hard-coded to <code>true</code> .
<code>String getCatalog()</code>	3.0	Yes	
Properties <code>getClientInfo()</code>	4.0	Yes	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
String getClientInfo(String name)	4.0	Yes	
int getHoldability()	3.0	Yes	Hard-coded to <code>CLOSE_CURSORS_AT_COMMIT</code> .
DatabaseMetaData getMetaData()	3.0	Yes	
int getNetworkTimeout()	4.1	No	
String getSchema()	4.1	Yes	The returned schema name does not always match the one used by statements. Statements use the schema name defined in the connection URL.
int getTransactionIsolation()	3.0	Yes	Hard-coded to <code>TRANSACTION_READ_UNCOMMITTED</code> .
Map<String,Class<?>> getTypeMap()	3.0	No	
SQLWarning getWarnings()	3.0	Yes	
boolean isClosed()	3.0	Yes	
boolean isReadOnly()	3.0	Yes	Returns <code>true</code> .
boolean isValid(int timeout)	4.0	Yes	
String nativeSQL(String sql)	3.0	Yes	
CallableStatement prepareCall(String sql)	3.0	No	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
CallableStatement prepareCall (String sql, int resultSetType, int resultSetConcurrency)	3.0	No	
CallableStatement prepareCall (String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)	3.0	No	
PreparedStatement prepareStatement(String sql)	3.0	Yes	
PreparedStatement prepareStatement(String sql, int autoGeneratedKeys)	3.0	No	
PreparedStatement prepareStatement(String sql, int[] columnIndexes)	3.0	No	
PreparedStatement prepareStatement(String sql, int resultSetType, int resultSetConcurrency)	3.0	No	
PreparedStatement prepareStatement(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)	3.0	No	
PreparedStatement prepareStatement(String sql, String [] columnNames)	3.0	No	
void releaseSavepoint(Savepoint savepoint)	3.0	No	Savepoints are not available because transactions are not supported.

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void rollback()</code>	3.0	No	Savepoints are not available because transactions are not supported.
<code>void rollback(Savepoint savepoint)</code>	3.0	No	Savepoints are not available because transactions are not supported.
<code>void setAutoCommit(boolean autoCommit)</code>	3.0	Yes	Ignored because auto-commit is hard-coded to <code>true</code> .
<code>void setCatalog(String catalog)</code>	3.0	Yes	
<code>void setClientInfo(Properties properties)</code>	4.0	Yes	
<code>void setClientInfo(String name, String value)</code>	4.0	Yes	
<code>void setHoldability(int holdability)</code>	3.0	Yes	
<code>void setNetworkTimeout(Executor executor, int milliseconds)</code>	4.1	No	
<code>void setReadOnly(boolean readOnly)</code>	3.0	Yes	
<code>Savepoint setSavepoint()</code>	3.0	No	Savepoints are not available because transactions are not supported.
<code>Savepoint setSavepoint(String name)</code>	3.0	No	Savepoints are not available because transactions are not supported.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void setSchema(String schema)</code>	4.1	Yes	Does not actually change the schema name used by newly created statements; only changes the value returned by <code>getSchema()</code> .
<code>void setTransactionIsolation(int level)</code>	3.0	Yes	
<code>void setTypeMap(Map<String, Class<?>> map)</code>	3.0	No	
<code>boolean isWrapperFor(Class<?> iface)</code>	4.0	Yes	
<code><T> T unwrap(Class<T> iface)</code>	4.0	Yes	

DatabaseMetaData

The following table lists the methods that belong to the `DatabaseMetaData` interface, and describes whether each method is supported by the Cloudera JDBC Connector for Apache Hive and which version of the JDBC API is the earliest version that supports the method.

For detailed information about each method in the `DatabaseMetaData` interface, see the Java API documentation: <http://docs.oracle.com/javase/1.5.0/docs/api/java/sql/DatabaseMetaData.html>.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>boolean allProceduresAreCallable()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean allTablesAreSelectable()</code>	3.0	Yes	Returns <code>true</code> .

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>boolean autoCommitFailureClosesAllResultSets()</code>	4.0	Yes	Returns <code>true</code> .
<code>boolean dataDefinitionCausesTransactionCommit()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean dataDefinitionIgnoredInTransactions()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean deletesAreDetected(int type)</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean doesMaxRowSizeIncludeBlobs()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean generatedKeyAlwaysReturned()</code>	4.1	Yes	
<code>ResultSet getAttributes(String catalog, String schemaPattern, String typeNamePattern, String attributeNamePattern)</code>	3.0	Yes	
<code>ResultSet getBestRowIdentifier(String catalog, String schema, String table, int scope, boolean nullable)</code>	3.0	Yes	
<code>ResultSet getCatalogs()</code>	3.0	Yes	
<code>String getCatalogSeparator()</code>	3.0	Yes	
<code>String getCatalogTerm()</code>	3.0	Yes	
<code>ResultSet getClientInfoProperties()</code>	4.0	Yes	
<code>ResultSet getColumnPrivileges(String catalog, String schema, String table, String columnNamePattern)</code>	3.0	Yes	
<code>ResultSet getColumns(String catalog, String schemaPattern, String tableNamePattern, String columnNamePattern)</code>	3.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
Connection getConnection()	3.0	Yes	
ResultSet getCrossReference(String primaryCatalog, String primarySchema, String primaryTable, String foreignCatalog, String foreignSchema, String foreignTable)	3.0	Yes	
int getDatabaseMajorVersion()	3.0	Yes	
int getDatabaseMinorVersion()	3.0	Yes	
String getDatabaseProductName()	3.0	Yes	Hard-coded to Impala.
String getDatabaseProductVersion()	3.0	Yes	
int getDefaultTransactionIsolation()	3.0	Yes	Hard-coded to TRANSACTION_READ_UNCOMMITTED.
int getDriverMajorVersion()	3.0	Yes	
int getDriverMinorVersion()	3.0	Yes	
String getDriverName()	3.0	Yes	Hard-coded to ImpalaJDBC.
String getDriverVersion()	3.0	Yes	
ResultSet getExportedKeys(String catalog, String schema, String table)	3.0	Yes	
String getExtraNameCharacters()	3.0	Yes	Returns an empty String.

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
ResultSet getFunctionColumns(String catalog, String schemaPattern, String functionNamePattern, String columnNamePattern)	4.0	Yes	
ResultSet getFunctions(String catalog, String schemaPattern, String functionNamePattern)	4.0	Yes	
String getIdentifierQuoteString()	3.0	Yes	Returns a backquote (`)
ResultSet getImportedKeys(String catalog, String schema, String table)	3.0	Yes	
ResultSet getIndexInfo(String catalog, String schema, String table, boolean unique, boolean approximate)	3.0	Yes	
int getJDBCMajorVersion()	3.0	Yes	
int getJDBCMajorVersion()	3.0	Yes	
int getMaxBinaryLiteralLength()	3.0	Yes	Returns 0.
int getMaxCatalogNameLength()	3.0	Yes	Returns 128.
int getMaxCharLiteralLength()	3.0	Yes	Returns 0.
int getMaxColumnNameLength()	3.0	Yes	Returns 128.
int getMaxColumnsInGroupBy()	3.0	Yes	Returns 0.
int getMaxColumnsInIndex()	3.0	Yes	Returns 0.
int getMaxColumnsInOrderBy()	3.0	Yes	Returns 0.
int getMaxColumnsInSelect()	3.0	Yes	Returns 0.
int getMaxColumnsInTable()	3.0	Yes	Returns 0.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>int getMaxConnections()</code>	3.0	Yes	Returns 0.
<code>int getMaxCursorNameLength()</code>	3.0	Yes	Returns 0.
<code>int getMaxIndexLength()</code>	3.0	Yes	Returns 0.
<code>int getMaxProcedureNameLength()</code>	3.0	Yes	Returns 0.
<code>int getMaxRowSize()</code>	3.0	Yes	Returns 0.
<code>int getMaxSchemaNameLength()</code>	3.0	Yes	Returns 128.
<code>int getMaxStatementLength()</code>	3.0	Yes	Returns 0.
<code>int getMaxStatements()</code>	3.0	Yes	Returns 0.
<code>int getMaxTableNameLength()</code>	3.0	Yes	Returns 128.
<code>int getMaxTablesInSelect()</code>	3.0	Yes	Returns 0.
<code>int getMaxUserNameLength()</code>	3.0	Yes	Returns 0.
<code>String getNumericFunctions()</code>	3.0	Yes	Returns the Numeric Functions list from the specification related to the JDBC version of the connector.
<code>ResultSet getPrimaryKeys(String catalog, String schema, String table)</code>	3.0	Yes	
<code>ResultSet getProcedureColumns(String catalog, String schemaPattern, String procedureNamePattern, String columnNamePattern)</code>	3.0	Yes	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
ResultSet getProcedures(String catalog, String schemaPattern, String procedureNamePattern)	3.0	Yes	
String getProcedureTerm()	3.0	Yes	Returns procedure.
ResultSet getPseudoColumns(String catalog, String schemaPattern, String tableNamePattern, String columnNamePattern)	4.1	Yes	
int getResultSetHoldability()	3.0	Yes	Returns CLOSE_CURSORS_AT_COMMIT.
RowIdLifetime getRowIdLifetime()	4.0	Yes	Returns ROWID_UNSUPPORTED.
ResultSet getSchemas()	3.0	Yes	
ResultSet getSchemas(String catalog, String schemaPattern)	4.0	Yes	
String getSchemaTerm()	3.0	Yes	Returns schema.
String getSearchStringEscape()	3.0	Yes	Returns a backslash (\).
String getSQLKeywords()	3.0	Yes	Returns an empty String.
int getSQLStateType()	3.0	Yes	Returns sqlStateSQL99.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
String getStringFunctions()	3.0	Yes	Returns the String Functions list from the specification related to the JDBC version of the connector.
ResultSet getSuperTables(String catalog, String schemaPattern, String tableNamePattern)	3.0	Yes	
ResultSet getSuperTypes(String catalog, String schemaPattern, String typeNamePattern)	3.0	Yes	
String getSystemFunctions()	3.0	Yes	Returns DATABASE, IFNULL, USER.
ResultSet getTablePrivileges(String catalog, String schemaPattern, String tableNamePattern)	3.0	Yes	
ResultSet getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)	3.0	Yes	
ResultSet getTableTypes()	3.0	Yes	
String getTimeDateFunctions()	3.0	Yes	Returns the Time and Date Functions list from the specification related to the JDBC version of the connector.
ResultSet getTypeInfo()	3.0	Yes	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
ResultSet getUDTs(String catalog, String schemaPattern, String typeNamePattern, int[] types)	3.0	Yes	
String getURL()	3.0	Yes	
String getUsername()	3.0	Yes	
ResultSet getVersionColumns(String catalog, String schema, String table)	3.0	Yes	
boolean insertsAreDetected(int type)	3.0	Yes	
boolean isCatalogAtStart()	3.0	Yes	
boolean isReadOnly()	3.0	Yes	Returns <code>true</code> .
boolean locatorsUpdateCopy()	3.0	Yes	Returns <code>false</code> .
boolean nullPlusNonNullIsNull()	3.0	Yes	Returns <code>true</code> .
boolean nullsAreSortedAtEnd()	3.0	Yes	Returns <code>false</code> .
boolean nullsAreSortedAtStart()	3.0	Yes	Returns <code>false</code> .
boolean nullsAreSortedHigh()	3.0	Yes	Returns <code>false</code> .
boolean nullsAreSortedLow()	3.0	Yes	Returns <code>true</code> .
boolean othersDeletesAreVisible(int type)	3.0	Yes	
boolean othersInsertsAreVisible(int type)	3.0	Yes	
boolean othersUpdatesAreVisible(int type)	3.0	Yes	
boolean ownDeletesAreVisible(int type)	3.0	Yes	
boolean ownInsertsAreVisible(int type)	3.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>boolean ownUpdatesAreVisible(int type)</code>	3.0	Yes	
<code>boolean storesLowerCaseIdentifiers()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean storesLowerCaseQuotedIdentifiers()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean storesMixedCaseIdentifiers()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean storesMixedCaseQuotedIdentifiers()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean storesUpperCaseIdentifiers()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean storesUpperCaseQuotedIdentifiers()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsAlterTableWithAddColumn()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsAlterTableWithDropColumn()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsANSI92EntryLevelSQL()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsANSI92FullSQL()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsANSI92IntermediateSQL()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsBatchUpdates()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsCatalogsInDataManipulation()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsCatalogsInIndexDefinitions()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsCatalogsInPrivilegeDefinitions()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsCatalogsInProcedureCalls()</code>	3.0	Yes	Returns <code>true</code> .

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>boolean supportsCatalogsInTableDefinitions()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsColumnAliasing()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsConvert()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsConvert(int fromType, int toType)</code>	3.0	Yes	
<code>boolean supportsCoreSQLGrammar()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsCorrelatedSubqueries()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsDataDefinitionAndDataManipulationTransactions()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsDataManipulationTransactionsOnly()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsDifferentTableCorrelationNames()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsExpressionsInOrderBy()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsExtendedSQLGrammar()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsFullOuterJoins()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsGetGeneratedKeys()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsGroupBy()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsGroupByBeyondSelect()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsGroupByUnrelated()</code>	3.0	Yes	Returns <code>false</code> .

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>boolean supportsIntegrityEnhancementFacility()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsLikeEscapeClause()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsLimitedOuterJoins()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsMinimumSQLGrammar()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsMixedCaseIdentifiers()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsMixedCaseQuotedIdentifiers()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsMultipleOpenResults()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsMultipleResultSets()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsMultipleTransactions()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsNamedParameters()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsNonNullableColumns()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsOpenCursorsAcrossCommit()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsOpenCursorsAcrossRollback()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsOpenStatementsAcrossCommit()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsOpenStatementsAcrossRollback()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsOrderByUnrelated()</code>	3.0	Yes	Returns <code>false</code> .

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>boolean supportsOuterJoins()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsPositionedDelete()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsPositionedUpdate()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsResultSetConcurrency(int type, int concurrency)</code>	3.0	Yes	
<code>boolean supportsResultSetHoldability(int holdability)</code>	3.0	Yes	
<code>boolean supportsResultSetType(int type)</code>	3.0	Yes	
<code>boolean supportsSavepoints()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsSchemasInDataManipulation()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsSchemasInIndexDefinitions()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsSchemasInPrivilegeDefinitions()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsSchemasInProcedureCalls()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsSchemasInTableDefinitions()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsSelectForUpdate()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsStatementPooling()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsStoredFunctionsUsingCallSyntax()</code>	4.0	Yes	Returns <code>false</code> .
<code>boolean supportsStoredProcedures()</code>	3.0	Yes	Returns <code>true</code> .

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>boolean supportsSubqueriesInComparisons()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsSubqueriesInExists()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsSubqueriesInIns()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsSubqueriesInQuantifieds()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsTableCorrelationNames()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsTransactionIsolationLevel(int level)</code>	3.0	Yes	
<code>boolean supportsTransactions()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean supportsUnion()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean supportsUnionAll()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean updatesAreDetected(int type)</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean usesLocalFilePerTable()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean usesLocalFiles()</code>	3.0	Yes	Returns <code>false</code> .
<code>boolean isWrapperFor(Class<?> iface)</code>	4.0	Yes	
<code><T> T unwrap(Class<T> iface)</code>	4.0	Yes	

DataSource

The following table lists the methods that belong to the `DataSource` interface, and describes whether each method is supported by the Cloudera JDBC Connector for Apache Hive and which version of the JDBC API is the earliest version that supports the method.

For detailed information about each method in the `DataSource` interface, see the Java API documentation: <http://docs.oracle.com/javase/1.5.0/docs/api/javax/sql/DataSource.html>.

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
Connection getConnection()	3.0	Yes	
Connection getConnection (String username, String password)	3.0	Yes	
int getLoginTimeout()	3.0	Yes	
PrintWriter getLogWriter()	3.0	Yes	
Logger getParentLogger()	4.1	No	The connector does not use <code>java.util.logging</code> .
void setLoginTimeout(int seconds)	3.0	Yes	
void setLogWriter(PrintWriter out)	3.0	Yes	
boolean isWrapperFor (Class<?> iface)	4.0	Yes	
<T> T unwrap(Class<T> iface)	4.0	Yes	

Driver

The following table lists the methods that belong to the `Driver` interface, and describes whether each method is supported by the Cloudera JDBC Connector for Apache Hive and which version of the JDBC API is the earliest version that supports the method.

For detailed information about each method in the `Driver` interface, see the Java API documentation: <http://docs.oracle.com/javase/1.5.0/docs/api/java/sql/Driver.html>.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
boolean acceptsURL(String url)	3.0	Yes	
Connection connect(String url, Properties info)	3.0	Yes	
int getMajorVersion()	3.0	Yes	
int getMinorVersion()	3.0	Yes	
Logger getParentLogger()	4.1	No	
DriverPropertyInfo[] getPropertyInfo(String url, Properties info)	3.0	Yes	
boolean jdbcCompliant()	3.0	Yes	

ParameterMetaData

The following table lists the methods that belong to the `ParameterMetaData` interface, and describes whether each method is supported by the Cloudera JDBC Connector for Apache Hive and which version of the JDBC API is the earliest version that supports the method.

For detailed information about each method in the `ParameterMetaData` interface, see the Java API documentation: <http://docs.oracle.com/javase/1.5.0/docs/api/java/sql/ParameterMetaData.html>.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
String getParameterClassName(int param)	3.0	Yes	
int getParameterCount()	3.0	Yes	
int getParameterMode(int param)	3.0	Yes	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>int getParameterType(int param)</code>	3.0	Yes	
<code>String getParameterTypeName(int param)</code>	3.0	Yes	
<code>int getPrecision(int param)</code>	3.0	Yes	
<code>int getScale(int param)</code>	3.0	Yes	
<code>int isNullable(int param)</code>	3.0	Yes	
<code>boolean isSigned(int param)</code>	3.0	Yes	
<code>boolean isWrapperFor(Class<?> iface)</code>	4.0	Yes	
<code><T> T unwrap(Class<T> iface)</code>	4.0	Yes	

PooledConnection

The following table lists the methods that belong to the `PooledConnection` interface, and describes whether each method is supported by the Cloudera JDBC Connector for Apache Hive and which version of the JDBC API is the earliest version that supports the method.

For detailed information about each method in the `PooledConnection` interface, see the Java API documentation: <http://docs.oracle.com/javase/1.5.0/docs/api/javax/sql/PooledConnection.html>.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void addConnectionEventListener(ConnectionEventListener listener)</code>	3.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void addStatementEventListener (StatementEventListener listener)</code>	4.0	Yes	
<code>void close()</code>	3.0	Yes	
<code>Connection getConnection()</code>	3.0	Yes	
<code>void removeConnectionEventListener (ConnectionEventListener listener)</code>	3.0	Yes	
<code>void removeStatementEventListener (StatementEventListener listener)</code>	4.0	Yes	Removes the specified <code>StatementEventListener</code> from the list of components that will be notified when the connector detects that a <code>PreparedStatement</code> has been closed or is invalid.

PreparedStatement

The `PreparedStatement` interface extends the `Statement` interface.

The following table lists the methods that belong to the `PreparedStatement` interface, and describes whether each method is supported by the Cloudera JDBC Connector for Apache Hive and which version of the JDBC API is the earliest version that supports the method.

For detailed information about each method in the `PooledConnection` interface, see the Java API documentation: <http://docs.oracle.com/javase/1.5.0/docs/api/java/sql/PreparedStatement.html>.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void addBatch()</code>	3.0	Yes	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void clearParameters()</code>	3.0	Yes	
<code>boolean execute()</code>	3.0	Yes	
<code>ResultSet executeQuery()</code>	3.0	Yes	
<code>int executeUpdate()</code>	3.0	Yes	If an updated row count is not available from the server, the connector returns a row count of -1.
<code>ResultSetMetaData getMetaData()</code>	3.0	Yes	
<code>ParameterMetaData getParameterMetaData()</code>	3.0	Yes	
<code>void setArray(int parameterIndex, Array x)</code>	3.0	No	
<code>void setAsciiStream(int parameterIndex, InputStream x)</code>	4.0	Yes	
<code>void setAsciiStream(int parameterIndex, InputStream x, int length)</code>	3.0	Yes	
<code>void setAsciiStream(int parameterIndex, InputStream x, long length)</code>	4.0	Yes	
<code>void setBigDecimal(int parameterIndex, BigDecimal x)</code>	3.0	Yes	
<code>void setBinaryStream(int parameterIndex, InputStream x)</code>	4.0	Yes	
<code>void setBinaryStream(int parameterIndex, InputStream x, int length)</code>	3.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void setBinaryStream(int parameterIndex, InputStream x, long length)</code>	4.0	Yes	
<code>void setBlob(int parameterIndex, Blob x)</code>	3.0	No	
<code>void setBlob(int parameterIndex, InputStream inputStream)</code>	4.0	No	
<code>void setBlob(int parameterIndex, InputStream inputStream, long length)</code>	4.0	No	
<code>void setBoolean(int parameterIndex, boolean x)</code>	3.0	Yes	
<code>void setByte(int parameterIndex, byte x)</code>	3.0	Yes	
<code>void setBytes(int parameterIndex, byte[] x)</code>	3.0	Yes	
<code>void setCharacterStream(int parameterIndex, Reader reader)</code>	4.0	Yes	
<code>void setCharacterStream(int parameterIndex, Reader reader, int length)</code>	3.0	Yes	
<code>void setCharacterStream(int parameterIndex, Reader reader, long length)</code>	4.0	Yes	
<code>void setClob(int parameterIndex, Clob x)</code>	3.0	No	
<code>void setClob(int parameterIndex, Reader reader)</code>	4.0	No	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void setClob(int parameterIndex, Reader reader, long length)</code>	4.0	No	
<code>void setDate(int parameterIndex, Date x)</code>	3.0	Yes	
<code>void setDate(int parameterIndex, Date x, Calendar cal)</code>	3.0	Yes	
<code>void setDouble(int parameterIndex, double x)</code>	3.0	Yes	
<code>void setFloat(int parameterIndex, float x)</code>	3.0	Yes	
<code>void setInt(int parameterIndex, int x)</code>	3.0	Yes	
<code>void setLong(int parameterIndex, long x)</code>	3.0	Yes	
<code>void setNCharacterStream(int parameterIndex, Reader value)</code>	4.0	No	
<code>void setNCharacterStream(int parameterIndex, Reader value, long length)</code>	4.0	No	
<code>void setNClob(int parameterIndex, NClob value)</code>	4.0	No	
<code>void setNClob(int parameterIndex, Reader reader)</code>	4.0	No	
<code>void setNClob(int parameterIndex, Reader reader, long length)</code>	4.0	No	
<code>void setNString(int parameterIndex, String value)</code>	4.0	No	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void setNull(int paramIndex, int sqlType, String typeName)</code>	3.0	Yes	
<code>void setObject(int parameterIndex, Object x)</code>	3.0	Yes	
<code>void setObject(int parameterIndex, Object x, int targetSqlType)</code>	3.0	Yes	
<code>void setObject(int parameterIndex, Object x, int targetSqlType, int scale)</code>	3.0	Yes	
<code>void setRef(int parameterIndex, Ref x)</code>	3.0	No	
<code>void setRowId(int parameterIndex, RowId x)</code>	4.0	No	
<code>void setShort(int parameterIndex, short x)</code>	3.0	No	
<code>void setSQLXML(int parameterIndex, SQLXML xmlObject)</code>	4.0	Yes	
<code>void setString(int parameterIndex, String x)</code>	3.0	Yes	
<code>void setTime(int parameterIndex, Time x)</code>	3.0	Yes	
<code>void setTime(int parameterIndex, Time x, Calendar cal)</code>	3.0	Yes	
<code>void setTimestamp(int parameterIndex, Timestamp x)</code>	3.0	Yes	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void setTimestamp(int parameterIndex, Timestamp x, Calendar cal)</code>	3.0	Yes	
<code>void setUnicodeStream(int parameterIndex, InputStream x, int length)</code>	3.0	Yes	Deprecated.
<code>void setURL(int parameterIndex, URL x)</code>	3.0	No	
<code>boolean isWrapperFor(Class<?> iface)</code>	4.0	Yes	
<code><T> T unwrap(Class<T> iface)</code>	4.0	Yes	

ResultSet

The following table lists the methods that belong to the `ResultSet` interface, and describes whether each method is supported by the Cloudera JDBC Connector for Apache Hive and which version of the JDBC API is the earliest version that supports the method.

For detailed information about each method in the `ResultSet` interface, see the Java API documentation: <http://docs.oracle.com/javase/1.5.0/docs/api/java/sql/ResultSet.html>.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>boolean absolute(int row)</code>	3.0	No	
<code>void afterLast()</code>	3.0	No	
<code>void beforeFirst()</code>	3.0	No	
<code>void cancelRowUpdates()</code>	3.0	No	Not valid because the connector is read-only.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void clearWarnings()</code>	3.0	Yes	
<code>void close()</code>	3.0	Yes	
<code>void deleteRow()</code>	3.0	No	Not valid because the connector is read-only.
<code>int findColumn(String columnName)</code>	3.0	Yes	
<code>boolean first()</code>	3.0	No	
<code>Array getArray(int i)</code>	3.0	No	
<code>Array getArray(String colName)</code>	3.0	No	
<code>InputStream getAsciiStream(int columnIndex)</code>	3.0	Yes	
<code>InputStream getAsciiStream(String columnName)</code>	3.0	Yes	
<code>BigDecimal getBigDecimal(int columnIndex)</code>	3.0	Yes	
<code>BigDecimal getBigDecimal(int columnIndex, int scale)</code>	3.0	Yes	Deprecated.
<code>BigDecimal getBigDecimal(String columnName)</code>	3.0	Yes	
<code>BigDecimal getBigDecimal(String columnName, int scale)</code>	3.0	Yes	Deprecated.
<code>InputStream getBinaryStream(int columnIndex)</code>	3.0	Yes	
<code>InputStream getBinaryStream(String columnName)</code>	3.0	Yes	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
Blob getBlob(int i)	3.0	No	
Blob getBlob(String colName)	3.0	No	
boolean getBoolean(int columnIndex)	3.0	Yes	
boolean getBoolean(String columnName)	3.0	Yes	
getByte(int columnIndex)	3.0	Yes	
byte getByte(String columnName)	3.0	Yes	
byte[] getBytes(int columnIndex)	3.0	Yes	
byte[] getBytes(String columnName)	3.0	Yes	
Reader getCharacterStream(int columnIndex)	3.0	Yes	
Reader getCharacterStream (String columnName)	3.0	Yes	
Clob getClob(int i)	3.0	No	
Clob getClob(String colName)	3.0	No	
int getConcurrency()	3.0	Yes	
String getCursorName()	3.0	Yes	
Date getDate(int columnIndex)	3.0	Yes	
Date getDate(int columnIndex, Calendar cal)	3.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
Date getDate(String columnName)	3.0	Yes	
Date getDate(String columnName, Calendar cal)	3.0	Yes	
double getDouble(int columnIndex)	3.0	Yes	
double getDouble(String columnName)	3.0	Yes	
int getFetchDirection()	3.0	Yes	
int getFetchSize()	3.0	Yes	
float getFloat(int columnIndex)	3.0	Yes	
float getFloat(String columnName)	3.0	Yes	
int getHoldability()	4.0	Yes	
int getInt(int columnIndex)	3.0	Yes	
int getInt(String columnName)	3.0	Yes	
long getLong(int columnIndex)	3.0	Yes	
long getLong(String columnName)	3.0	Yes	
ResultSetMetaData getMetaData()	3.0	Yes	
Reader getNCharacterStream(int columnIndex)	4.0	No	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
Reader getNCharacterStream(String columnLabel)	4.0	No	
NClob getNClob(int columnIndex)	4.0	No	
NClob getNClob(String columnLabel)	4.0	No	
String getNString(int columnIndex)	4.0	No	
String getNString(String columnLabel)	4.0	No	
Object getObject(int columnIndex)	3.0	Yes	
<T> T getObject(int columnIndex, Class<T> type)	4.1	No	
Object getObject(int i, Map<String,Class<?>> map)	3.0	No	
Object getObject(String columnName)	3.0	No	
<T> T getObject(String columnName, Class<T> type)	4.1	No	
Object getObject(String colName, Map<String,Class<?>> map)	3.0	Yes	
Ref getRef(int i)	3.0	No	
Ref getRef(String colName)	3.0	No	
int getRow()	3.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
RowId getRowId(int columnIndex)	4.0	No	
RowId getRowId(String columnLabel)	4.0	No	
short getShort(int columnIndex)	3.0	Yes	
short getShort(String columnName)	3.0	Yes	
SQLXML getSQLXML(int columnIndex)	4.0	No	
SQLXML getSQLXML(String columnLabel)	4.0	No	
Statement getStatement()	3.0	Yes	
String getString(int columnIndex)	3.0	Yes	
String getString(String columnName)	3.0	Yes	
Time getTime(int columnIndex)	3.0	Yes	
Time getTime(int columnIndex, Calendar cal)	3.0	Yes	
Time getTime(String columnName)	3.0	Yes	
Time getTime(String columnName, Calendar cal)	3.0	Yes	
Timestamp getTimestamp(int columnIndex)	3.0	Yes	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
Timestamp getTimestamp(int columnIndex, Calendar cal)	3.0	Yes	
Timestamp getTimestamp(String columnName)	3.0	Yes	
Timestamp getTimestamp(String columnName, Calendar cal)	3.0	Yes	
int getType()	3.0	Yes	
InputStream getUnicodeStream(int columnIndex)	3.0	Yes	Deprecated.
InputStream getUnicodeStream(String columnName)	3.0	Yes	Deprecated.
URL getURL(int columnIndex)	3.0	No	
URL getURL(String columnName)	3.0	No	
SQLWarning getWarnings()	3.0	Yes	
void insertRow()	3.0	No	Not valid because the connector is read-only.
boolean isAfterLast()	3.0	Yes	
boolean isBeforeFirst()	3.0	Yes	
boolean isClosed()	4.0	Yes	
boolean isFirst()	3.0	Yes	
boolean isLast()	3.0	No	
boolean last()	3.0	No	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void moveToCurrentRow()</code>	3.0	No	Not valid because the connector is read-only.
<code>void moveToInsertRow()</code>	3.0	No	Not valid because the connector is read-only.
<code>boolean next()</code>	3.0	Yes	
<code>boolean previous()</code>	3.0	No	
<code>void refreshRow()</code>	3.0	No	
<code>boolean relative(int rows)</code>	3.0	No	
<code>boolean rowDeleted()</code>	3.0	Yes	Hard-coded to <code>false</code> .
<code>boolean rowInserted()</code>	3.0	Yes	Hard-coded to <code>false</code> .
<code>boolean rowUpdated()</code>	3.0	Yes	Hard-coded to <code>false</code> .
<code>void setFetchDirection(int direction)</code>	3.0	No	Not valid because the connector is forward-only.
<code>void setFetchSize(int rows)</code>	3.0	Yes	
<code>void updateArray(int columnIndex, Array x)</code>	3.0	No	
<code>void updateArray(String columnName, Array x)</code>	3.0	No	
<code>void updateAsciiStream(int columnIndex, InputStream x)</code>	4.0	No	Not valid because the connector is read-only.
<code>void updateAsciiStream(int columnIndex, InputStream x, int length)</code>	3.0	No	Not valid because the connector is read-only.

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void updateAsciiStream(int columnIndex, InputStream x, long length)</code>	4.0	No	Not valid because the connector is read-only.
<code>void updateAsciiStream(String columnName, InputStream x)</code>	4.0	No	Not valid because the connector is read-only.
<code>void updateAsciiStream(String columnName, InputStream x, int length)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateAsciiStream(String columnName, InputStream x, long length)</code>	4.0	No	Not valid because the connector is read-only.
<code>void updateBigDecimal(int columnIndex, BigDecimal x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateBigDecimal(String columnName, BigDecimal x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateBinaryStream(int columnIndex, InputStream x)</code>	4.0	No	Not valid because the connector is read-only.
<code>void updateBinaryStream(int columnIndex, InputStream x, int length)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateBinaryStream(int columnIndex, InputStream x, long length)</code>	4.0	No	Not valid because the connector is read-only.
<code>void updateBinaryStream(String columnName, InputStream x)</code>	4.0	No	Not valid because the connector is read-only.
<code>void updateBinaryStream(String columnName, InputStream x, int length)</code>	3.0	No	Not valid because the connector is read-only.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void updateBinaryStream(String columnName, InputStream x, long length)</code>	4.0	No	Not valid because the connector is read-only.
<code>void updateBlob(int columnIndex, InputStream inputStream)</code>	4.0	No	
<code>void updateBlob(int columnIndex, Blob x)</code>	3.0	No	
<code>void updateBlob(int columnIndex, InputStream inputStream, long length)</code>	4.0	No	
<code>void updateBlob(String columnName, InputStream inputStream)</code>	4.0	No	
<code>void updateBlob(String columnName, Blob x)</code>	3.0	No	
<code>void updateBlob(String columnLabel, InputStream inputStream, long length)</code>	4.0	No	
<code>void updateBoolean(int columnIndex, boolean x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateBoolean(String columnName, boolean x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateByte(int columnIndex, byte x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateByte(String columnName, byte x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateBytes(int columnIndex, byte[] x)</code>	3.0	No	Not valid because the connector is read-only.

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void updateBytes(String columnName, byte[] x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateCharacterStream(int columnIndex, Reader x, int length)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateCharacterStream(String columnName, Reader reader, int length)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateBlob(int columnIndex, InputStream inputStream)</code>	4.0	No	
<code>void updateClob(int columnIndex, Clob x)</code>	3.0	No	
<code>void updateBlob(int columnIndex, InputStream inputStream, long length)</code>	4.0	No	
<code>void updateBlob(String columnName, InputStream inputStream)</code>	4.0	No	
<code>void updateClob(String columnName, Clob x)</code>	3.0	No	
<code>void updateBlob(String columnName, InputStream inputStream, long length)</code>	4.0	No	
<code>void updateDate(int columnIndex, Date x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateDate(String columnName, Date x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateDouble(int columnIndex, double x)</code>	3.0	No	Not valid because the connector is read-only.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void updateDouble(String columnName, double x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateFloat(int columnIndex, float x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateFloat(String columnName, float x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateInt(int columnIndex, int x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateInt(String columnName, int x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateLong(int columnIndex, long x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateLong(String columnName, long x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateNCharacterStream(int columnIndex, Reader x)</code>	4.0	No	
<code>void updateNCharacterStream(int columnIndex, Reader x, long length)</code>	4.0	No	
<code>void updateNCharacterStream(String columnName, Reader reader)</code>	4.0	No	
<code>void updateNCharacterStream(String columnName, Reader reader, long length)</code>	4.0	No	
<code>void updateNClob(int columnIndex, NClob nClob)</code>	4.0	No	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void updateNClob(int columnIndex, Reader reader)</code>	4.0	No	
<code>void updateNClob(int columnIndex, Reader reader, long length)</code>	4.0	No	
<code>void updateNClob(String columnName, NClob nClob)</code>	4.0	No	
<code>void updateNClob(String columnName, Reader reader)</code>	4.0	No	
<code>void updateNClob(String columnName, Reader reader, long length)</code>	4.0	No	
<code>void updateNString(int columnIndex, String nString)</code>	4.0	No	
<code>void updateNString(String columnName, String nString)</code>	4.0	No	
<code>void updateNull(int columnIndex)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateNull(String columnName)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateObject(int columnIndex, Object x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateObject(int columnIndex, Object x, int scale)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateObject(String columnName, Object x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateObject(String columnName, Object x, int scale)</code>	3.0	No	Not valid because the connector is read-only.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void updateRef(int columnIndex, Ref x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateRef(String columnName, Ref x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateRow()</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateRowId(int columnIndex, RowId x)</code>	4.0	No	
<code>void updateRowId(String columnName, RowId x)</code>	4.0	No	
<code>void updateShort(int columnIndex, short x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateShort(String columnName, short x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateSQLXML(int columnIndex, SQLXML xmlObject)</code>	4.0	No	
<code>void updateSQLXML(String columnName, SQLXML xmlObject)</code>	4.0	No	
<code>void updateString(int columnIndex, String x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateString(String columnName, String x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateTime(int columnIndex, Time x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateTime(String columnName, Time x)</code>	3.0	No	Not valid because the connector is read-only.

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void updateTimestamp(int columnIndex, Timestamp x)</code>	3.0	No	Not valid because the connector is read-only.
<code>void updateTimestamp(String columnName, Timestamp x)</code>	3.0	No	Not valid because the connector is read-only.
<code>boolean wasNull()</code>	3.0	Yes	
<code>boolean isWrapperFor(Class<?> iface)</code>	4.0	Yes	
<code><T> T unwrap(Class<T> iface)</code>	4.0	Yes	

ResultSetMetaData

The following table lists the methods that belong to the `ResultSetMetaData` interface, and describes whether each method is supported by the Cloudera JDBC Connector for Apache Hive and which version of the JDBC API is the earliest version that supports the method.

For detailed information about each method in the `ResultSetMetaData` interface, see the Java API documentation: <http://docs.oracle.com/javase/1.5.0/docs/api/java/sql/ResultSetMetaData.html>.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>String getCatalogName(int column)</code>	3.0	Yes	
<code>String getColumnClassName(int column)</code>	3.0	Yes	
<code>int getColumnCount()</code>	3.0	Yes	
<code>int getColumnDisplaySize(int column)</code>	3.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
String getColumnLabel(int column)	3.0	Yes	
String getColumnName(int column)	3.0	Yes	
int getColumnType(int column)	3.0	Yes	
String getColumnTypeName(int column)	3.0	Yes	
int getPrecision(int column)	3.0	Yes	
int getScale(int column)	3.0	Yes	
String getSchemaName(int column)	3.0	Yes	
String getTableName(int column)	3.0	Yes	
boolean isAutoIncrement(int column)	3.0	Yes	
boolean isCaseSensitive(int column)	3.0	Yes	
boolean isCurrency(int column)	3.0	Yes	
boolean isDefinitelyWritable(int column)	3.0	Yes	
int isNullable(int column)	3.0	Yes	
boolean isReadOnly(int column)	3.0	Yes	
boolean isSearchable(int column)	3.0	Yes	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
boolean isSigned(int column)	3.0	Yes	
boolean isWritable(int column)	3.0	Yes	
boolean isWrapperFor(Class<?> iface)	4.0	Yes	
<T> T unwrap(Class<T> iface)	4.0	Yes	

Statement

The following table lists the methods that belong to the `Statement` interface, and describes whether each method is supported by the Cloudera JDBC Connector for Apache Hive and which version of the JDBC API is the earliest version that supports the method.

For detailed information about each method in the `Statement` interface, see the Java API documentation: <http://docs.oracle.com/javase/1.5.0/docs/api/java/sql/Statement.html>.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
void addBatch(String sql)	3.0	Yes	
void cancel()	3.0	Yes	
void clearBatch()	3.0	Yes	
void clearWarnings()	3.0	Yes	
void close()	3.0	Yes	
void closeOnCompletion()	4.1	Yes	
boolean execute(String sql)	3.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>boolean execute(String sql, int autoGeneratedKeys)</code>	3.0	No	
<code>boolean execute(String sql, int[] columnIndexes)</code>	3.0	No	
<code>boolean execute(String sql, String[] columnNames)</code>	3.0	No	
<code>int[] executeBatch()</code>	3.0	No	
<code>ResultSet executeQuery(String sql)</code>	3.0	Yes	
<code>int executeUpdate(String sql)</code>	3.0	Yes	If an updated row count is not available from the server, the connector returns a row count of -1.
<code>int executeUpdate(String sql, int autoGeneratedKeys)</code>	3.0	No	If an updated row count is not available from the server, the connector returns a row count of -1.
<code>int executeUpdate(String sql, int[] columnIndexes)</code>	3.0	No	If an updated row count is not available from the server, the connector returns a row count of -1.
<code>int executeUpdate(String sql, String[] columnNames)</code>	3.0	No	If an updated row count is not available from the server, the connector returns a row count of -1.
<code>Connection getConnection()</code>	3.0	Yes	
<code>int getFetchDirection()</code>	3.0	Yes	

Features

Method	Supported Since JDBC Version	Supported by the Connector	Notes
int getFetchSize()	3.0	Yes	
ResultSet getGeneratedKeys()	3.0	Yes	
int getMaxFieldSize()	3.0	Yes	
int getMaxRows()	3.0	Yes	
boolean getMoreResults()	3.0	Yes	
boolean getMoreResults(int current)	3.0	No	
int getQueryTimeout()	3.0	Yes	
ResultSet getResultSet()	3.0	Yes	
int getResultSetConcurrency()	3.0	Yes	Hard-coded to CONCUR_READ_ONLY.
int getResultSetHoldability()	3.0	Yes	Hard-coded to CLOSE_CURSORS_AT_COMMIT.
int getResultSetType()	3.0	Yes	Hard-coded to TYPE_FORWARD_ONLY.
int getUpdateCount()	3.0	Yes	
SQLWarning getWarnings()	3.0	Yes	
boolean isClosed()	4.0	Yes	
boolean isCloseOnCompletion()	4.1	Yes	
boolean isPoolable()	4.0	Yes	
void setCursorName(String name)	3.0	No	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void setEscapeProcessing(boolean enable)</code>	3.0	Yes	
<code>void setFetchDirection(int direction)</code>	3.0	No	
<code>void setFetchSize(int rows)</code>	3.0	Yes	
<code>void setMaxFieldSize(int max)</code>	3.0	Yes	
<code>void setMaxRows(int max)</code>	3.0	Yes	
<code>void setPoolable(boolean poolable)</code>	4.0	Yes	
<code>void setQueryTimeout(int seconds)</code>	3.0	Yes	
<code>boolean isWrapperFor(Class<?> iface)</code>	4.0	Yes	
<code><T> T unwrap(Class<T> iface)</code>	4.0	Yes	

Connector Configuration Options

Connector Configuration Options lists and describes the properties that you can use to configure the behavior of the Cloudera JDBC Connector for Apache Hive.

You can set configuration properties using the connection URL. For more information, see [Building the Connection URL](#).

Note:
Property names and values are case-sensitive.

AllowSelfSignedCerts

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

- 1: The connector allows self-signed certificates.

Important:
When this property is set to 1, SSL verification is disabled. The connector does not verify the server certificate against the trust store, and does not verify if the server's host name matches the common name in the server certificate.

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

Note:
This property is applicable only when SSL connections are enabled.

Default Value	Data Type	Required
0	Integer	No

AsyncExecPollInterval

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 Hive is asynchronous. It does not mean that JDBC asynchronous operations are supported.

Note:
This option is applicable only to HDInsight clusters

Default Value	Data Type	Required
10	Integer	No

AuthMech

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.
- 6 for Hadoop Delegation Token.
- 12 for Single Sign-On
- 14 for JWT

Default Value	Data Type	Required
Depends on the <code>transportMode</code> setting. For more information, see TransportMode .	Integer	No

BinaryColumnLength

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

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

Default Value	Data Type	Required
32767	Integer	No

CAIssuedCertsMismatch

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

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



Note:

This property is applicable only when SSL connections are enabled.

Default Value	Data Type	Required
0	Integer	No

CatalogSchemaSwitch

This property specifies whether the connector treats Hive catalogs as schemas or as catalogs.

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

Default Value	Data Type	Required
0	Integer	No

DecimalColumnScale

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

Default Value	Data Type	Required
10	Integer	No

DefaultStringColumnLength

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 Hive does not specify a maximum data length for STRING columns.

Default Value	Data Type	Required
255	Integer	No

DelegationToken

A Hadoop delegation token for authentication.

This token must be provided to the connector in the form of a Base64 URL-safe encoded string. It can be obtained from the connector using the `getDelegationToken()` function, or by utilizing the Hadoop distribution `.jar` files.

Default Value	Data Type	Required
None	String	Yes, if <code>AuthMech</code> is set to 6 (Hadoop Delegation Token)

DelegationUID

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



Note:

This option is applicable only when connecting to a Hive Server 2 instance that supports this feature.

Default Value	Data Type	Required
None	String	No

EnableServerLogs

This property controls whether to enable the printing of the server logs to the driver logs.

- 1: To enable the printing of the server logs.
- 0: To disable the printing of the server logs .

Default Value	Data Type	Required
0	Boolean	No

FastConnection

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

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

Default Value	Data Type	Required
0	Integer	No

httpPath

The partial URL corresponding to the Hive server.

The connector 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:hive2://localhost:10002;AuthMech=3;transportMode=http;
httpPath=cliservice;UID=jsmith;PWD=cloudera123;
```

**Note:**

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

Default Value	Data Type	Required
None	String	Yes, if <code>transportMode=http</code> .

JWTString

The JSON Web Token used to access the server.

Default Value	Data Type	Required
None	String	Yes, if <code>AuthMech=14</code> (JWT).

IgnoreTransactions

This property specifies whether the connector ignores transaction-related operations or returns an error.

- 1: The connector ignores any transaction related operations and returns success.
- 0: The connector returns an "operation not supported" error if it attempts to run a query that contains transaction related operations.

Default Value	Data Type	Required
0	Boolean	No

KrbAuthType

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

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

- 1: The connector checks the `java.security.auth.login.config` system property for a JAAS configuration. If a JAAS configuration is specified, the connector uses that information to create a `LoginContext` and then uses the Subject associated with it.
- 2: The connector checks the `KRB5_CONFIG` and `KRB5CCNAME` system environment variables for a Kerberos ticket cache. The connector uses the information from the cache to create a `LoginContext` and then uses the Subject associated with it.
- 3: The connector uses the native GSS-API feature in the JDK to use the Kerberos tickets in the native Windows credentials cache without the need to set the `AllowTgtSessionKey` property in the Windows registry.

**Note:**

- The `Native GSS-API` feature is only available in Java 11 or later. While Java 13 and later include a default `Native GSS-API` library. While a default `Native GSS-API` library might be included in a future version of Java 11, if you are using Java 11 it does not include a default `Native GSS-API` library, you may work around the issue by setting the `sun.security.jgss.lib` system property to point to a `sspi_bridge.dll` file included in Java 13 or higher.
- JAAS configuration is disabled by default. To enable JAAS configuration, please set the `JDBC_ENABLE_JAAS` environment variable to 1.

Below is an example of setting the `sun.security.jgss.lib` system property in the Java start-up command to point to the default native GSS-API library included in Java 13.

```
-Dsun.security.jgss.lib="C:\Program Files\Java\jdk-13.0.2\bin\sspi_bridge.dll"
```

Default Value	Data Type	Required
0	Integer	No

KrbHostFQDN

The fully qualified domain name of the Hive Server 2 host.

Default Value	Data Type	Required
None	String	Yes, if <code>AuthMech=1</code> .

KrbRealm

The realm of the Hive Server 2 host.

If your Kerberos configuration already defines the realm of the Hive Server 2 host as the default realm, then you do not need to configure this property.

Default Value	Data Type	Required
Depends on your Kerberos configuration	String	No

KrbServiceName

The Kerberos service principal name of the Hive server.

Default Value	Data Type	Required
None	String	Yes, if <code>AuthMech=1</code> .

LoginTimeout

The maximum time in seconds that the connector waits when attempting to first establish connection with the database.

When this property is set to 0, connections do not time out.



Note:

- When both `LoginTimeout` and `SocketTimeout` are provided, the smaller value is utilized during the initial connection.
- When the value is both provided through `DriverManager.setLoginTimeout()` and a connection property, the connection property value takes precedence.

Default Value	Data Type	Required
0	Integer	No

LogLevel

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.

The settings for logging apply to every connection that uses the Cloudera JDBC Connector for Apache Hive, 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 connector to abort.
- 2: Enable logging on the ERROR level, which logs error events that might still allow the connector 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 connector.
- 5: Enable logging on the DEBUG level, which logs detailed information that is useful for debugging the connector.
- 6: Enable logging on the TRACE level, which logs all connector activity.

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

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

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

Default Value	Data Type	Required
0	Integer	No

LogPath

The full path to the folder where the connector 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.

Default Value	Data Type	Required
The current working directory	String	No

NonRowcountQueryPrefixes

A comma-separated list of queries used to return a regular result set, rather than a row count.

For example:

```
NonRowcountQueryPrefixes=INSERT,UPDATE,DELETE;
```

Default Value	Data Type	Required
None	String	No

PreparedMetaLimitZero

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`.

Default Value	Data Type	Required
0	Integer	No

PWD

The password corresponding to the user name that you provided using the property [UID](#).



Important:

If you set the `AuthMech` to 3, the default `PWD` value is not used and you must specify a password.

Default Value	Data Type	Required
anonymous	String	Yes, if <code>AuthMech=3</code> .

RowsFetchedPerBlock

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.

Default Value	Data Type	Required
10000	Integer	No

ServerLogsPoolInterval

This property indicates the interval between server requests of retrieving logs in milliseconds.

Default Value	Data Type	Required
100	Integer	No

SocketTimeout

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.

Default Value	Data Type	Required
0	Integer	No

SSL

This property specifies whether the connector communicates with the Hive server through an SSL-enabled socket.

- 1: The connector connects to SSL-enabled sockets.
- 2: The connector connects to SSL-enabled sockets using two-way authentication.
- 0: The connector does not connect to SSL-enabled sockets.



Note:

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

Default Value	Data Type	Required
0	Integer	No

SSLKeyStore

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

See also the property [SSLKeyStorePwd](#).



Note:

The Cloudera JDBC Connector for Apache Hive accepts TrustStores and KeyStores for one-way SSL authentication. See also the property [SSLTrustStore](#).

Default Value	Data Type	Required
None	String	No

SSLKeyStoreProvider

Default Value	Data Type	Required
None	String	No

Description

The provider of the Java Security API for the KeyStore that is being used for one-way SSL authentication.

SSLKeyStorePwd

The password for accessing the Java KeyStore that you specified using the property [SSLKeyStore](#).

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

SSLKeyStoreType

Default Value	Data Type	Required
JKS	String	No

Description

The type of Java KeyStore that is being used for one-way SSL authentication.

SSLTrustStore

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

If the trust store requires a password, provide it using the property [SSLTrustStorePwd](#). See [SSLTrustStorePwd](#).



Note:

The Cloudera JDBC Connector for Apache Hive accepts TrustStores and KeyStores for one-way SSL authentication. See also the property [SSLKeyStore](#).

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

SSLTrustStoreProvider

Default Value	Data Type	Required
None	String	No

Description

The provider of the Java Security API for the TrustStore that is being used for one-way SSL authentication.

SSLTrustStorePwd

The password for accessing the Java TrustStore that you specified using the property [SSLTrustStore](#).

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

SSLTrustStoreType

Default Value	Data Type	Required
JKS	String	No

Description

The type of Java TrustStore that is being used for one-way SSL authentication.

TransportMode

The transport protocol to use in the Thrift layer.

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

Connector Configuration Options

When connecting to a Hive Server 1 instance, you must use this setting. If you use this setting and do not specify the `AuthMech` property, then the connector uses `AuthMech=0` by default. This setting is valid only when the `AuthMech` property is set to 0 or 3.

- `sasl`: The connector uses the SASL transport protocol.

If you use this setting but do not specify the `AuthMech` property, then the connector uses `AuthMech=2` by default. This setting is valid only when the `AuthMech` property is set to 1, 2, or 3.

- `http`: The connector uses the HTTP transport protocol.

If you use this setting but do not specify the `AuthMech` property, then the connector 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](#).

Default Value	Data Type	Required
<code>sasl</code>	String	No

UID

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



Important:

If you set the `AuthMech` to 3, the default `UID` value is not used and you must specify a user name.

Default Value	Data Type	Required
<code>anonymous</code>	String	Yes, if <code>AuthMech=3</code> .

UseNativeQuery

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

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



Note:

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

Default Value	Data Type	Required
0	Integer	No

zk

The connection string to one or more ZooKeeper quorums, written in the following format where *[ZK_IP]* is the IP address, *[ZK_Port]* is the port number, and *[ZK_Namespace]* is the namespace:

[ZK_IP]:[ZK_Port]/[ZK_Namespace]

For example:

`jdbc:hive2://zk=192.168.0.1:2181/hiveserver2`

Use this option to enable the Dynamic Service Discovery feature, which allows you to connect to Hive servers that are registered against a ZooKeeper service by connecting to the ZooKeeper service.

You can specify multiple quorums in a comma-separated list. If connection to a quorum fails, the connector will attempt to connect to the next quorum in the list.

Default Value	Data Type	Required
None	String	No

Contact Us

If you are having difficulties using the connector, our [Community Forum](#) may have your solution. In addition to providing user to user support, our forums are a great place to share your questions, comments, and feature requests with us.

If you are a Subscription customer you may also use the [Cloudera Support Portal](#) to search the Knowledge Base or file a Case.



Important:

To help us assist you, prior to contacting Cloudera Support please prepare a detailed summary of the client and server environment including operating system version, patch level, and configuration.

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 Hive, Apache, and Hive 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.