



Cloudera JDBC Connector for Apache Impala

Important Notice

© 2010-2026 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: January 2026

Contents

ABOUT THE CLUDERA JDBC CONNECTOR FOR APACHE IMPALA	7
SYSTEM REQUIREMENTS	8
CLOUDERA JDBC CONNECTOR FOR APACHE IMPALA FILES	9
INSTALLING AND USING THE CLUDERA JDBC CONNECTOR FOR APACHE IMPALA	10
REFERENCING THE JDBC CONNECTOR LIBRARIES	10
<i>Using the Connector in a JDBC Application</i>	10
<i>Using the Connector in Java Code</i>	10
REGISTERING THE CONNECTOR CLASS	11
BUILDING THE CONNECTION URL	12
CONFIGURING AUTHENTICATION	14
USING NO AUTHENTICATION	14
USING KERBEROS	14
USING JSON WEB TOKEN (JWT)	15
USING USER NAME	16
USING USER NAME AND PASSWORD	16
USING SINGLE SIGN-ON	17
USING OAUTH 2.0	17
<i>Token Pass-through</i>	17
CONFIGURING KERBEROS AUTHENTICATION FOR WINDOWS	18
<i>Downloading and Installing MIT Kerberos for Windows</i>	19
<i>Using the MIT Kerberos Ticket Manager to Get Tickets</i>	19
Setting the KRB5CCNAME Environment Variable	19
Getting a Kerberos Ticket	20
Authenticating to the Impala Server	20
<i>Using the Connector to Get Tickets</i>	20

Deleting the KRB5CCNAME Environment Variable	20
Setting Up the Kerberos Configuration File	21
Setting Up the JAAS Login Configuration File	21
Authenticating to the Impala Server	21
<i>Using an Existing Subject to Authenticate the Connection</i>	22
USING KERBEROS CONSTRAINED DELEGATION	23
CONFIGURING SSL	24
CONFIGURING SERVER-SIDE PROPERTIES	26
CONFIGURING LOGGING	27
FEATURES	29
SQL TRANSLATION	29
DATA TYPES	29
CATALOG AND SCHEMA SUPPORT	30
WRITE-BACK	30
SECURITY AND AUTHENTICATION	30
MULTITHREADING SUPPORT	31
INTERFACES AND SUPPORTED METHODS	31
<i>CallableStatement</i>	32
<i>Connection</i>	38
<i>DatabaseMetaData</i>	41
<i>DataSource</i>	49
<i>Driver</i>	50
<i>ParameterMetaData</i>	50
<i>PooledConnection</i>	51
<i>PreparedStatement</i>	52
<i>ResultSet</i>	55
<i>ResultSetMetaData</i>	64
<i>Statement</i>	66
CONNECTOR CONFIGURATION OPTIONS	68
ALLOWSELSIGNEDCERTS	68

ASYNCEXEC POLLINTERVAL	68
AUTH_ACCESS_TOKEN	68
AUTH_FLOW	68
AUTHMECH	69
AUTORECONNECT	69
CAISSUEDCERTSMISMATCH	69
CATALOGSCHEMASWITCH	70
DATACONVERSIONINFETCHTHREAD	70
DEFAULTSTRINGCOLUMNLENGTH	70
DELEGATIONUID	70
ENABLEMULTISOCKETFETCH	71
HTTPPATH	71
JWTSTRING	71
IGNORETRANSACTIONS	71
KRBAUTHTYPE	71
KRBHOSTFQDN	72
KRBREALM	73
KRBSERVICENAME	73
LOGLEVEL	73
LOGPATH	74
LOWERCASERESULTSETCOLUMNNAME	74
MAXNUMMULTISOCKETFETCHHANDLERS	74
MULTISOCKETFETCHROWSETPEROPERATION	74
NONROWCOUNTQUERYPREFIXES	75
OPTIMIZEDINSERT	75
PREPAREDMETALIMITZERO	75

PWD	75
ROWSFETCHEDPERBLOCK	76
SOCKETTIMEOUT	76
SSL	76
SSLKEYSTORE	76
SSLKEYSTOREPROVIDER	77
SSLKEYSTOREPWD	77
SSLKEYSTORETYPE	77
SSLTRUSTSTOREPROVIDER	77
SSLTRUSTSTORE	77
SSLTRUSTSTOREPWD	78
SSLTRUSTSTORETYPE	78
SSOWEBSEVERTIMEOUT	78
STRIPCATALOGNAME	78
SUPPORTTIMEONLYTIMESTAMP	78
TRANSPORTMODE	79
UID	79
UPPERCASERESULTSETCOLNAME	79
USENATIVEQUERY	79
USESASL (DEPRECATED)	80
CONTACT US	81

About the Cloudera JDBC Connector for Apache Impala

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

The Cloudera JDBC Connector for Apache Impala 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 Impala 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 connector's release notes.

Cloudera JDBC Connector for Apache Impala Files

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

- `ImpalaJDBC41_[Version].zip`
- `ImpalaJDBC42_[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 Impala

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

To access an Impala data store using the Cloudera JDBC Connector for Apache Impala, 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)

The Cloudera JDBC Connector for Apache Impala is a forward-only, read-only connector with no transaction support. Because the connector does not support transactions, auto-commit is always set to **true**.

Referencing the JDBC Connector Libraries

Before you use the Cloudera JDBC Connector for Apache Impala, 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 7:

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

For Java SE 8:

- For Windows:
<http://docs.oracle.com/javase/8/docs/technotes/tools/windows/classpath.html>
- For Linux and Solaris:
<http://docs.oracle.com/javase/8/docs/technotes/tools/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 Impala to Impaladata 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.impala.jdbc.Driver`
- `com.cloudera.impala.jdbc.DataSource`

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

- `com.cloudera..jdbc.Driver`
- `com.cloudera..jdbc.DataSource`

To support JDBC 4.2, classes with the following fully-qualified class names (FQCNs) are available:

- `com.cloudera..jdbc42.Driver`
- `com.cloudera..jdbc42.DataSource`

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;
    Class.forName(DRIVER_CLASS);
    connection = DriverManager.getConnection(CONNECTION_URL);
    return connection;
}
```

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

```
private static Connection connectViaDS() throws Exception
{

```

```

Connection connection = null;
Class.forName(DRIVER_CLASS);
DataSource ds = new com.cloudera.impala.jdbc.DataSource();
com.cloudera.impala.jdbc.DataSource();
ds.setURL(CONNECTION_URL);
connection = ds.getConnection();
return connection;
}

```

Building the Connection URL

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

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

By default, the connector uses port 28000 when `TransportMode` is set to `http`, and 21050 when `TransportMode` is not set or is set to `sasl` or `binary`.

By default, the connector uses the schema named **default**.

You can specify optional settings such as 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](#).

If you specify a property that is not supported by the connector, then the connector attempts to apply the property as a Impala 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:impala://[Host]:[Port]/[Schema];[Property1]=[Value];
[Property2]=[Value];...
```

For example, to connect to port 18000 on an Impala server installed on the local machine, use a schema named `default2`, and authenticate the connection using a user name and password, you would use the following connection URL:

```
jdbc:impala://node1.example.com:18000/default2;AuthMech=3;
UID=cloudera;PWD=cloudera
```

- Properties are case-sensitive.
- Do not duplicate properties in the connection URL.
- Property values containing semicolons must use the URL-encoded format %3B.

For example: `PWD=simba%3Bsimba`, the connector interprets the password as *simba;simba*.

Configuring Authentication

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

- No Authentication
- Kerberos
- JWT
- User Name
- User Name And Password
- Single Sign-On (SSO)
- Using OAuth 2.0

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

For information about configuring the authentication mechanism that Impala uses, see the Impala documentation: <http://www.cloudera.com/content/cloudera/en/documentation.html>.

For information about the properties you can use in the connection URL, see Connector Configuration Options.

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:

Set the `AuthMech` property to 0.

For example:

```
jdbc:impala://localhost:21050;AuthMech=0;
```

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.

The connector also supports Kerberos constrained delegation. For more details on this, see Using Kerberos Constrained Delegation .

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 Impala server is not the default, then set the `KrbRealm` property to the realm of the Impala server.
3. Set the `KrbHostFQDN` property to the fully qualified domain name of the Impala server host.
4. If you are using Kerberos Constrained Delegation, set the `userGSSCredential` property to your Kerberos GSS Credential.
5. 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 Impala server with Kerberos enabled, but without SSL enabled:

```
jdbc:impala://node1.example.com:21050;AuthMech=1;
KrbRealm=EXAMPLE.COM;KrbHostFQDN=node1.example.com;
KrbServiceName=impala
```

In this example, Kerberos is enabled for JDBC connections, the Kerberos service principal name is `impala/node1.example.com@EXAMPLE.COM`, the host name for the data source is `node1.example.com`, and the server is listening on port 21050 for JDBC connections.

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](#).

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:impala  
://node1.example.com:21050;AuthMech=14;JWTString=s0m3t0ken5tr1ngh  
3re
```

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](#).

To configure User Name authentication:

1. Set the `AuthMech` property to 2.
2. Set the `UID` property to an appropriate user name for accessing the Impala server.

For example:

```
jdbc:impala://node1.example.com:21050;AuthMech=2;UID=impala
```

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](#).

To configure User Name And Password authentication:

1. Set the `AuthMech` property to 3.
2. Set the `UID` property to an appropriate user name for accessing the Impala server.
3. Set the `PWD` property to the password corresponding to the user name you provided.

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

```
jdbc:impala://node1.example.com:21050;AuthMech=3;  
UID=impala;PWD=cloudera;  
jdbc:impala://node1.example.com:10000;AuthMech=3;  
transportMode=http;httpPath=cliservice;UID=impala;PWD=cloudera;
```


In this example, user name and password authentication is enabled for JDBC connections, the LDAP user name is `impala`, the password is `cloudera`, and the server is listening on port 21050 for JDBC connections.

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

Using Single Sign-On

Single Sign-On (SSO) is a process that allows network users to access all authorized network resources without having to log in to each resource separately. For example, implementing SSO for users within an organization allows each user to authenticate to Impala without providing a separate set of Impala credentials.

You specify the properties in the connection URL. For more information about the syntax of the connection URL, see [Building the Connection URL](#).

SSL is required for this authentication method. For more information, see [Configuring SSL](#).

To configure Single Sign-On authentication:

1. Set the `AuthMech` property to `12`.
2. Set the `TransportMode` property to `http`.
3. Optionally, set the `SSOWebServerTimeout` property to the number of seconds that the connector waits before timing out while waiting for a browser response.

For example:

```
jdbc:impala://node1.example.com:28000;AuthMech=12;
SSL=1;TransportMode=http;httpPath=cliservice;SSOWebServerTimeout=
60;
```

Using OAuth 2.0

Three types of authentication work flows are available when using OAuth 2.0: Token pass-through, Client credentials, and Browser based authentication.

When you use OAuth 2.0 authentication, HTTP is the only Thrift transport protocol available. Browser based authentication work flow only works when SSL is enabled.

Token Pass-through

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

To configure OAuth 2.0 token pass-through authentication :

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

Using Client Credentials Based

This authentication mechanism requires SSL to be enabled.

To configure OAuth 2.0 Client Credentials authentication:

1. Set the `AuthMech` property to 11.
2. Set the `Auth_Flow` property to 1.
3. Set the `OAuth2ClientId` property to client id used in specified authentication source.
4. Set the `OAuth2Secret` property to secret used in the specified authentication source.
5. Set the `OAuth2Audience` property to audience used in the specified authentication source.
6. Set the `OAuth2ConnAuthTokenEndpoint` property to the token endpoint used in the specified authentication source.

Using Authorization Code Based Authentication

To configure OAuth 2.0 Authorization Code based authentication:

1. Set the `AuthMech` property to 11.
2. Set the `Auth_Flow` property to 2.
3. Set the `OAuth2ClientId` property to client id used in specified authentication source.
4. Set the `OAuth2Secret` property to secret used in the specified authentication source.
5. Set the `OAuth2Audience` property to audience used in the specified authentication source.
6. Set the `OAuth2ConnAuthTokenEndpoint` property to the token endpoint used in the specified authentication source.
7. Set the `OAuth2ConnAuthAuthorizeEndpoint` property to the authorization endpoint used in the specified authentication source.

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>.

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 Impala 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 Impala 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 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**.

- 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:

- Create a standard `krb5.ini` file and place it in the `C:\Windows` directory.
- 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:

- 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@CLOUDERA"
  doNotPrompt=true;
};
```

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

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

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

Authenticating to the Impala 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 Impala 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
    static String ConnectionURL =
    "jdbc:impala://192.168.1.1:21050";

    /**
     * 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;
```

Using Kerberos Constrained Delegation

The connector can also be configured to use Kerberos Constrained Delegation. This feature allows a service to obtain service tickets to a restricted list of other services running on specific servers on the network after it has been presented with a service ticket. For more details on the process see: <https://technet.microsoft.com/en-ca/library/cc995228.aspx>.

The `userGSSCredential` connection property can be used in the connection URL to pass in a `GSSCredential` object. The following sample code shows how to use the property to pass the `GSSCredential` into the connector using JDBC 4.1.

```
GSSCredential userCredential = [GSSCredential]
Driver driver = (Driver) Class.forName
("com.cloudera.impala.jdbc.Driver").newInstance();
Properties properties = new Properties();
properties.put("userGSSCredential", userCredential);
Connection conn = driver.connect(
    "jdbc:impala
    ://node1.example.com:21050;AuthMech=1;KrbRealm=EXAMPLE.CO
    M;
    KrbHostFQDN=node1.example.com;KrbServiceName=impala"
    ,properties);
```

Configuring SSL

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 an Impala 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:
 - i. Create a TrustStore containing your signed, trusted server certificate.
 - ii. Set the `SSLTrustStore` property to the full path of the TrustStore.
 - iii. Set the `SSLTrustStorePwd` property to the password for accessing the TrustStore.
 - iv. If the TrustStore is not a JKS TrustStore, set the `SSLTrustStoreType` property to the correct type. The supported types are:
 - i. `SSLTrustStoreType=BCFKS` (BouncyCastle FIPS Keystore)
 - ii. `SSLTrustStoreType=PKCS12` (Public Key Cryptography Standards #12)
`SSLTrustStoreType=PKCS11` (Public Key Cryptography Standards #11) TrustStore type is not supported.
 - v. 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:
 - a. Create a KeyStore containing your signed, trusted server certificate.
 - b. Set the `SSLKeyStore` property to the full path of the KeyStore.
 - c. Set the `SSLKeyStorePwd` property to the password for accessing the KeyStore.
 - d. If the KeyStore is not a JKS KeyStore, set the `SSLKeyStoreType` property to the correct type.
 - e. 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.

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 Impala 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:impala://localhost:21050;AuthMech=3;SSL=1;
SSLKeyStore=C:\\Users\\bsmith\\Desktop\\keystore.jks;SSLKeyStoreP
wd=clouderaSSL123;UID=impala;PWD=cloudera123
jdbc:impala://localhost:10000;AuthMech=3;SSL=1;
SSLKeyStore=C:\\Users\\bsmith\\Desktop\\keystore.jks;SSLKeyStoreP
wd=clouderaSSL123;UID=impala;PWD=cloudera123
```

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

Configuring Server-Side Properties

When connecting to a server that is running Impala 2.0 or later, you can use the connector to apply configuration properties to the server by setting the properties in the connection URL.

This feature is not supported for earlier versions of Impala, where the SET statement can only be executed from within the Impala shell.

For example, to set the `MEM_LIMIT` query option to 1 GB and the `REQUEST_POOL` query option to `myPool`, you would use a connection URL such as the following:

```
jdbc:impala://localhost:18000/default2;AuthMech=3;  
UID=cloudera;PWD=cloudera;MEM_LIMIT=1000000000;REQUEST_  
POOL=myPool
```

Configuring Logging

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

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 Impala, 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 Impala, 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:impala://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 Impala produces the following log files in the location specified in the `LogPath` property:

- An `ImpalaJDBC_driver.log` file that logs connector activity that is not specific to a connection.
- An `Impala_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 Impala:

- SQL Translation
- Data Types
- Catalog and Schema Support
- Write-back
- Security and Authentication
- Interfaces and Supported Methods
- Multithreading Support

SQL Translation

The Cloudera JDBC Connector for Apache Impala is able to parse queries locally prior to sending them to the Impala server. This feature allows the connector to calculate query metadata without executing the query, support query parameters, and support extra SQL features such as JDBC escape sequences and additional scalar functions that are not available in the Impala-shell tool.

The connector does not support translation for queries that reference a field contained in a nested column (an ARRAY, MAP, or STRUCT column). To retrieve data from a nested column, make sure that the query is written in valid Impala SQL syntax.

Data Types

The Cloudera JDBC Connector for Apache Impala supports many common data formats, converting between Impala, SQL, and Java data types.

The following table lists the supported data type mappings.

Impala Type	SQL Type	Java Type
ARRAY	VARCHAR	String
BIGINT	BIGINT	java.math.BigInteger
BINARY	VARBINARY	byte[]
BOOLEAN	BOOLEAN	Boolean
CHAR (Available only in CDH 5.2 or later)	CHAR	String
DATE	DATE	java.sql.Date
DECIMAL (Available only in CDH 5.1 or later)	DECIMAL	java.math.BigDecimal
DOUBLE	DOUBLE	Double

Impala Type	SQL Type	Java Type
(REAL is an alias for DOUBLE)		
FLOAT	REAL	Float
INT	INTEGER	Long
MAP	VARCHAR	String
SMALLINT	SMALLINT	Integer
STRUCT	VARCHAR	String
TIMESTAMP	TIMESTAMP	java.sql.Timestamp
TINYINT	TINYINT	Short
VARCHAR (Available only in CDH 5.2 or later)	VARCHAR	String

Catalog and Schema Support

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

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

Write-back

The Cloudera JDBC Connector for Apache Impala supports translation for the following syntax:

- INSERT
- CREATE
- DROP

The connector also supports translation for UPDATE and DELETE syntax, but only when querying Kudu tables while connected to an Impala server that is running Impala 2.7 or later.

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 Impala SQL.

Security and Authentication

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

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 Impala user name only, or your Impala user name and password. You must use the authentication mechanism that matches the security requirements of the Impala server. 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.

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.

Multithreading Support

The Cloudera JDBC Connector for Apache Impala supports multithreaded processing.

Within a given process, the connector enables multiple threads to access different connections concurrently. For each connection, the connector enables multiple threads to access different statements from the connection concurrently.

Interfaces and Supported Methods

The Cloudera JDBC Connector for Apache Impala 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	• SQLData
---	---

• Blob • Clob • Ref • Savepoint	• 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 Impala 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
<code>Array getArray(int i)</code>	3.0	No	
<code>Array getArray(String parameterName)</code>	3.0	No	
<code>BigDecimal getBigDecimal(int parameterIndex)</code>	3.0	Yes	
<code>BigDecimal getBigDecimal(int parameterIndex, int scale)</code>	3.0	Yes	Deprecated.
<code>BigDecimal getBigDecimal(String parameterName)</code>	3.0	Yes	
<code>Blob getBlob(int i)</code>	3.0	No	
<code>Blob getBlob(String parameterName)</code>	3.0	No	
<code>boolean getBoolean(int parameterIndex)</code>	3.0	Yes	
<code>boolean getBoolean(String parameterName)</code>	3.0	Yes	
<code>byte getByte(int parameterIndex)</code>	3.0	Yes	
<code>byte getByte(String parameterName)</code>	3.0	Yes	
<code>byte[] getBytes(int parameterIndex)</code>	3.0	Yes	
<code>byte[] getBytes(String parameterName)</code>	3.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
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	
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	

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	
<code>String getString(int parameterIndex)</code>	3.0	Yes	
<code>String getString(String parameterName)</code>	3.0	Yes	
<code>Time getTime(int parameterIndex)</code>	3.0	Yes	
<code>Time getTime(int parameterIndex, Calendar cal)</code>	3.0	Yes	
<code>Time getTime(String parameterName)</code>	3.0	Yes	
<code>Time getTime(String parameterName, Calendar cal)</code>	3.0	Yes	
<code>Timestamp getTimestamp(int</code>	3.0	Yes	

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

Method	Supported Since JDBC Version	Supported by the Connector	Notes
(String parameterName, InputStream x)			
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	
void setClob(String parameterName, Clob x)	4.0	Yes	
void setClob(String parameterName, Reader reader)	4.0	Yes	
void setClob(String parameterName, Reader reader, long length)	4.0	Yes	
void setDate(String parameterName, Date x)	3.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<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	
<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</code>	3.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
scale)			
void setRowId(String parameterName, RowId x)	4.0	Yes	
void setShort(String parameterName, short x)	3.0	Yes	
void setSQLXML(String parameterName, SQLXML xmlObject)	4.0	Yes	
void setString(String parameterName, String x)	3.0	Yes	
void setTime(String parameterName, Time x)	3.0	Yes	
void setTime(String parameterName, Time x, Calendar cal)	3.0	Yes	
void setTimestamp(String parameterName, Timestamp x)	3.0	Yes	
void setTimestamp(String parameterName, Timestamp x, Calendar cal)	3.0	Yes	
void setURL(String parameterName, URL val)	3.0	Yes	
boolean wasNull()	3.0	Yes	
boolean isWrapperFor(Class<?> iface)	4.0	Yes	
<T> T unwrap(Class<T> iface)	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 Impala 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
void clearWarnings()	3.0	Yes	
void close()	3.0	Yes	
void commit()	3.0	Yes	Auto-commit cannot be set to

Method	Supported Since JDBC Version	Supported by the Connector	Notes
			false because it is hard-coded to true.
Array createArrayOf(String typeName, Object[] elements)	4.0	No	
Blob createBlob()	4.0	No	
Clob createClob()	4.0	No	
NClob createNClob()	4.0	No	
SQLXML createSQLXML()	4.0	No	
Statement createStatement()	3.0	Yes	
Statement createStatement(int resultSetType, int resultSetConcurrency)	3.0	No	
Statement createStatement(int resultSetType, int resultSetConcurrency, int resultSetHoldability)	3.0	No	
Struct createStruct(String typeName, Object[] attributes)	4.0	No	
boolean getAutoCommit()	3.0	Yes	Hard-coded to true.
String getCatalog()	3.0	Yes	
Properties getClientInfo()	4.0	Yes	
String getClientInfo(String name)	4.0	Yes	
int getHoldability()	3.0	Yes	Hard-coded to CLOSE_CURSORS_AT_COMMIT.
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 TRANSACTION_READ_UNCOMMITTED.
Map<String, Class<?>> getTypeMap()	3.0	No	
SQLWarning getWarnings()	3.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>boolean isClosed()</code>	3.0	Yes	
<code>boolean isReadOnly()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean isValid(int timeout)</code>	4.0	Yes	
<code>String nativeSQL(String sql)</code>	3.0	Yes	
<code>CallableStatement prepareCall(String sql)</code>	3.0	No	
<code>CallableStatement prepareCall(String sql, int resultSetType, int resultSetConcurrency)</code>	3.0	No	
<code>CallableStatement prepareCall(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)</code>	3.0	No	
<code>PreparedStatement prepareStatement(String sql)</code>	3.0	Yes	
<code>PreparedStatement prepareStatement(String sql, int autoGeneratedKeys)</code>	3.0	No	
<code>PreparedStatement prepareStatement(String sql, int[] columnIndexes)</code>	3.0	No	
<code>PreparedStatement prepareStatement(String sql, int resultSetType, int resultSetConcurrency)</code>	3.0	No	
<code>PreparedStatement prepareStatement(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)</code>	3.0	No	
<code>PreparedStatement prepareStatement(String sql, String[] columnNames)</code>	3.0	No	
<code>void releaseSavepoint (Savepoint savepoint)</code>	3.0	No	Savepoints are not available because transactions are not supported.
<code>void rollback()</code>	3.0	No	Savepoints are not available because transactions are not supported.
<code>void rollback(Savepoint</code>	3.0	No	Savepoints are not available

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>savepoint()</code>			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.
<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 Impala 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 true.
<code>boolean allTablesAreSelectable()</code>	3.0	Yes	Returns true.
<code>boolean autoCommitFailureClosesAllResultSets()</code>	4.0	Yes	Returns true.
<code>boolean dataDefinitionCausesTransactionCommit()</code>	3.0	Yes	Returns false.
<code>boolean dataDefinitionIgnoredInTransactions()</code>	3.0	Yes	Returns false.
<code>boolean deletesAreDetected(int type)</code>	3.0	Yes	Returns true.
<code>boolean doesMaxRowSizeIncludeBlobs()</code>	3.0	Yes	Returns false.
<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	
<code>Connection getConnection()</code>	3.0	Yes	
<code>ResultSet getCrossReference(String primaryCatalog, String primarySchema, String primaryTable, String foreignCatalog, String foreignSchema, String foreignTable)</code>	3.0	Yes	
<code>int getDatabaseMajorVersion()</code>	3.0	Yes	
<code>int getDatabaseMinorVersion()</code>	3.0	Yes	
<code>String getDatabaseProductName()</code>	3.0	Yes	Hard-coded to

Method	Supported Since JDBC Version	Supported by the Connector	Notes
			Impala.
<code>String getDatabaseProductVersion()</code>	3.0	Yes	
<code>int getDefaultTransactionIsolation()</code>	3.0	Yes	Hard-coded to TRANSACTION_READ_UNCOMMITTED.
<code>int getDriverMajorVersion()</code>	3.0	Yes	
<code>int getDriverMinorVersion()</code>	3.0	Yes	
<code>String getDriverName()</code>	3.0	Yes	Hard-coded to ImpalaJDBC.
<code>String getDriverVersion()</code>	3.0	Yes	
<code>ResultSet getExportedKeys(String catalog, String schema, String table)</code>	3.0	Yes	
<code>String getExtraNameCharacters()</code>	3.0	Yes	Returns an empty String.
<code>ResultSet getFunctionColumns(String catalog, String schemaPattern, String functionNamePattern, String columnNamePattern)</code>	4.0	Yes	
<code>ResultSet getFunctions(String catalog, String schemaPattern, String functionNamePattern)</code>	4.0	Yes	
<code>String getIdentifierQuoteString()</code>	3.0	Yes	Returns a backquote (`)
<code>ResultSet getImportedKeys(String catalog, String schema, String table)</code>	3.0	Yes	
<code>ResultSet getIndexInfo(String catalog, String schema, String table, boolean unique, boolean approximate)</code>	3.0	Yes	
<code>int getJDBCMajorVersion()</code>	3.0	Yes	
<code>int getJDBCMinorVersion()</code>	3.0	Yes	
<code>int getMaxBinaryLiteralLength()</code>	3.0	Yes	Returns 0.
<code>int getMaxCatalogNameLength()</code>	3.0	Yes	Returns 128.
<code>int getMaxCharLiteralLength()</code>	3.0	Yes	Returns 0.
<code>int getMaxColumnNameLength()</code>	3.0	Yes	Returns 128.
<code>int getMaxColumnsInGroupBy()</code>	3.0	Yes	Returns 0.
<code>int getMaxColumnsInIndex()</code>	3.0	Yes	Returns 0.
<code>int getMaxColumnsInOrderBy()</code>	3.0	Yes	Returns 0.
<code>int getMaxColumnsInSelect()</code>	3.0	Yes	Returns 0.
<code>int getMaxColumnsInTable()</code>	3.0	Yes	Returns 0.
<code>int getMaxConnections()</code>	3.0	Yes	Returns 0.

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

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>String getSQLKeywords()</code>	3.0	Yes	Returns an empty String.
<code>int getSQLStateType()</code>	3.0	Yes	Returns <code>sqlStateSQL99</code> .
<code>String getStringFunctions()</code>	3.0	Yes	Returns the String Functions list from the specification related to the JDBC version of the connector.
<code>ResultSet getSuperTables(String catalog, String schemaPattern, String tableNamePattern)</code>	3.0	Yes	
<code>ResultSet getSuperTypes(String catalog, String schemaPattern, String typeNamePattern)</code>	3.0	Yes	
<code>String getSystemFunctions()</code>	3.0	Yes	Returns <code>DATABASE</code> , <code>IFNULL</code> , <code>USER</code> .
<code>ResultSet getTablePrivileges(String catalog, String schemaPattern, String tableNamePattern)</code>	3.0	Yes	
<code>ResultSet getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)</code>	3.0	Yes	
<code>ResultSet getTableTypes()</code>	3.0	Yes	
<code>String getTimeDateFunctions()</code>	3.0	Yes	Returns the Time and Date Functions list from the specification related to the JDBC version of the connector.
<code>ResultSet getTypeInfo()</code>	3.0	Yes	
<code>ResultSet getUDTs(String catalog, String schemaPattern, String typeNamePattern, int[] types)</code>	3.0	Yes	
<code>String getURL()</code>	3.0	Yes	
<code>String getUsername()</code>	3.0	Yes	
<code>ResultSet getVersionColumns(String catalog, String schema, String table)</code>	3.0	Yes	
<code>boolean insertsAreDetected(int type)</code>	3.0	Yes	
<code>boolean isCatalogAtStart()</code>	3.0	Yes	
<code>boolean isReadOnly()</code>	3.0	Yes	Returns <code>true</code> .
<code>boolean locatorsUpdateCopy()</code>	3.0	Yes	Returns <code>false</code> .

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

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>boolean supportsCatalogsInTableDefinitions()</code>	3.0	Yes	Returns true.
<code>boolean supportsColumnAliasing()</code>	3.0	Yes	Returns true.
<code>boolean supportsConvert()</code>	3.0	Yes	Returns true.
<code>boolean supportsConvert(int fromType, int toType)</code>	3.0	Yes	
<code>boolean supportsCoreSQLGrammar()</code>	3.0	Yes	Returns true.
<code>boolean supportsCorrelatedSubqueries()</code>	3.0	Yes	Returns true.
<code>boolean supportsDataDefinitionAndDataManipulationTransactions()</code>	3.0	Yes	Returns false.
<code>boolean supportsDataManipulationTransactionsOnly()</code>	3.0	Yes	Returns false.
<code>boolean supportsDifferentTableCorrelationNames()</code>	3.0	Yes	Returns false.
<code>boolean supportsExpressionsInOrderBy()</code>	3.0	Yes	Returns true.
<code>boolean supportsExtendedSQLGrammar()</code>	3.0	Yes	Returns false.
<code>boolean supportsFullOuterJoins()</code>	3.0	Yes	Returns true.
<code>boolean supportsGetGeneratedKeys()</code>	3.0	Yes	Returns false.
<code>boolean supportsGroupBy()</code>	3.0	Yes	Returns true.
<code>boolean supportsGroupByBeyondSelect()</code>	3.0	Yes	Returns true.
<code>boolean supportsGroupByUnrelated()</code>	3.0	Yes	Returns false.
<code>boolean supportsIntegrityEnhancementFacility()</code>	3.0	Yes	Returns false.
<code>boolean supportsLikeEscapeClause()</code>	3.0	Yes	Returns true.
<code>boolean supportsLimitedOuterJoins()</code>	3.0	Yes	Returns false.
<code>boolean supportsMinimumSQLGrammar()</code>	3.0	Yes	Returns true.
<code>boolean supportsMixedCaseIdentifiers()</code>	3.0	Yes	Returns false.
<code>boolean supportsMixedCaseQuotedIdentifiers()</code>	3.0	Yes	Returns true.
<code>boolean supportsMultipleOpenResults()</code>	3.0	Yes	Returns false.
<code>boolean supportsMultipleResultSets()</code>	3.0	Yes	Returns false.
<code>boolean supportsMultipleTransactions()</code>	3.0	Yes	Returns true.
<code>boolean supportsNamedParameters()</code>	3.0	Yes	Returns false.
<code>boolean supportsNonNullableColumns()</code>	3.0	Yes	Returns false.
<code>boolean supportsOpenCursorsAcrossCommit()</code>	3.0	Yes	Returns false.
<code>boolean</code>	3.0	Yes	Returns false.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>supportsOpenCursorsAcrossRollback()</code>			
<code>boolean supportsOpenStatementsAcrossCommit()</code>	3.0	Yes	Returns true.
<code>boolean supportsOpenStatementsAcrossRollback()</code>	3.0	Yes	Returns true.
<code>boolean supportsOrderByUnrelated()</code>	3.0	Yes	Returns false.
<code>boolean supportsOuterJoins()</code>	3.0	Yes	Returns false.
<code>boolean supportsPositionedDelete()</code>	3.0	Yes	Returns false.
<code>boolean supportsPositionedUpdate()</code>	3.0	Yes	Returns false.
<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 false.
<code>boolean supportsSchemasInDataManipulation()</code>	3.0	Yes	Returns true.
<code>boolean supportsSchemasInIndexDefinitions()</code>	3.0	Yes	Returns true.
<code>boolean supportsSchemasInPrivilegeDefinitions()</code>	3.0	Yes	Returns true.
<code>boolean supportsSchemasInProcedureCalls()</code>	3.0	Yes	Returns false.
<code>boolean supportsSchemasInTableDefinitions()</code>	3.0	Yes	Returns true.
<code>boolean supportsSelectForUpdate()</code>	3.0	Yes	Returns false.
<code>boolean supportsStatementPooling()</code>	3.0	Yes	Returns false.
<code>boolean supportsStoredFunctionsUsingCallSyntax()</code>	4.0	Yes	Returns false.
<code>boolean supportsStoredProcedures()</code>	3.0	Yes	Returns true.
<code>boolean supportsSubqueriesInComparisons()</code>	3.0	Yes	Returns true.
<code>boolean supportsSubqueriesInExists()</code>	3.0	Yes	Returns true.
<code>boolean supportsSubqueriesInIns()</code>	3.0	Yes	Returns true.
<code>boolean supportsSubqueriesInQuantifieds()</code>	3.0	Yes	Returns true.
<code>boolean supportsTableCorrelationNames()</code>	3.0	Yes	Returns true.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>boolean supportsTransactionIsolationLevel(int level)</code>	3.0	Yes	
<code>boolean supportsTransactions()</code>	3.0	Yes	Returns false.
<code>boolean supportsUnion()</code>	3.0	Yes	Returns true.
<code>boolean supportsUnionAll()</code>	3.0	Yes	Returns true.
<code>boolean updatesAreDetected(int type)</code>	3.0	Yes	Returns true.
<code>boolean usesLocalFilePerTable()</code>	3.0	Yes	Returns false.
<code>boolean usesLocalFiles()</code>	3.0	Yes	Returns false.
<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 Impala 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>.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>Connection getConnection()</code>	3.0	Yes	
<code>Connection getConnection(String username, String password)</code>	3.0	Yes	
<code>int getLoginTimeout()</code>	3.0	Yes	
<code>PrintWriter getLogWriter()</code>	3.0	Yes	
<code>Logger getParentLogger()</code>	4.1	No	The connector does not use <code>java.util.logging</code> .
<code>void setLoginTimeout(int seconds)</code>	3.0	Yes	
<code>void setLogWriter(PrintWriter out)</code>	3.0	Yes	
<code>boolean isWrapperFor(Class<?> iface)</code>	4.0	Yes	
<code><T> T unwrap(Class<T> iface)</code>	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 Impala 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
<code>boolean acceptsURL(String url)</code>	3.0	Yes	
<code>Connection connect(String url, Properties info)</code>	3.0	Yes	
<code>int getMajorVersion()</code>	3.0	Yes	
<code>int getMinorVersion()</code>	3.0	Yes	
<code>Logger getParentLogger()</code>	4.1	No	
<code>DriverPropertyInfo[] getPropertyInfo(String url, Properties info)</code>	3.0	Yes	
<code>boolean jdbcCompliant()</code>	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 Impala 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
<code>String getParameterClassName(int param)</code>	3.0	Yes	
<code>int getParameterCount()</code>	3.0	Yes	
<code>int getParameterMode(int param)</code>	3.0	Yes	
<code>int getParameterType(int param)</code>	3.0	Yes	
<code>String getParameterTypeName(int param)</code>	3.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<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 Impala 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	
<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 Impala 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	
<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	
<code>void setBinaryStream(int parameterIndex, InputStream x, long length)</code>	4.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>parameterIndex, InputStream x, long length)</code>			
<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	
<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	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<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	
<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</code>	3.0	Yes	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>parameterIndex, Time x)</code>			
<code>void setTime(int parameterIndex, Time x, Calendar cal)</code>	3.0	Yes	
<code>void setTimestamp(int parameterIndex, Timestamp x)</code>	3.0	Yes	
<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 Impala 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.
<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	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<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	
<code>Blob getBlob(int i)</code>	3.0	No	
<code>Blob getBlob(String colName)</code>	3.0	No	
<code>boolean getBoolean(int columnIndex)</code>	3.0	Yes	
<code>boolean getBoolean(String columnName)</code>	3.0	Yes	
<code>getByte(int columnIndex)</code>	3.0	Yes	
<code>byte getByte(String columnName)</code>	3.0	Yes	
<code>byte[] getBytes(int columnIndex)</code>	3.0	Yes	
<code>byte[] getBytes(String columnName)</code>	3.0	Yes	
<code>Reader getCharacterStream(int columnIndex)</code>	3.0	Yes	
<code>Reader getCharacterStream(String columnName)</code>	3.0	Yes	
<code>Clob getClob(int i)</code>	3.0	No	
<code>Clob getClob(String colName)</code>	3.0	No	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>int getConcurrency()</code>	3.0	Yes	
<code>String getCursorName()</code>	3.0	Yes	
<code>Date getDate(int columnIndex)</code>	3.0	Yes	
<code>Date getDate(int columnIndex, Calendar cal)</code>	3.0	Yes	
<code>Date getDate(String columnName)</code>	3.0	Yes	
<code>Date getDate(String columnName, Calendar cal)</code>	3.0	Yes	
<code>double getDouble(int columnIndex)</code>	3.0	Yes	
<code>double getDouble(String columnName)</code>	3.0	Yes	
<code>int getFetchDirection()</code>	3.0	Yes	
<code>int getFetchSize()</code>	3.0	Yes	
<code>float getFloat(int columnIndex)</code>	3.0	Yes	
<code>float getFloat(String columnName)</code>	3.0	Yes	
<code>int getHoldability()</code>	4.0	Yes	
<code>int getInt(int columnIndex)</code>	3.0	Yes	
<code>int getInt(String columnName)</code>	3.0	Yes	
<code>long getLong(int columnIndex)</code>	3.0	Yes	
<code>long getLong(String columnName)</code>	3.0	Yes	
<code>ResultSetMetaData getMetaData()</code>	3.0	Yes	
<code>Reader getNCharacterStream(int columnIndex)</code>	4.0	No	
<code>Reader getNCharacterStream(String columnLabel)</code>	4.0	No	
<code>NClob getNClob(int columnIndex)</code>	4.0	No	
<code>NClob getNClob(String columnLabel)</code>	4.0	No	
<code>String getNString(int columnIndex)</code>	4.0	No	
<code>String getNString(String</code>	4.0	No	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
columnLabel)			
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	
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	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
Timestamp getTimestamp(int columnIndex)	3.0	Yes	
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	
void moveToCurrentRow()	3.0	No	Not valid because the connector is read-only.
void moveToInsertRow()	3.0	No	Not valid because the connector is read-only.
boolean next()	3.0	Yes	
boolean previous()	3.0	No	
void refreshRow()	3.0	No	
boolean relative(int rows)	3.0	No	
boolean rowDeleted()	3.0	Yes	Hard-coded to false.
boolean rowInserted()	3.0	Yes	Hard-coded to false.
boolean rowUpdated()	3.0	Yes	Hard-coded to false.
void setFetchDirection(int direction)	3.0	No	Not valid because the connector is forward-only.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<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.
<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, long length)</code>	3.0	No	Not valid because the connector is read-only.

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>InputStream x, int length)</code>			
<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.
<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	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<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.
<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	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>void updateNClob(int columnIndex, NClob nClob)</code>	4.0	No	
<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.
<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</code>	4.0	No	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<code>columnName, RowId x)</code>			
<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.
<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 Impala 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	

Method	Supported Since JDBC Version	Supported by the Connector	Notes
<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	
<code>String getColumnLabel(int column)</code>	3.0	Yes	
<code>String getColumnName(int column)</code>	3.0	Yes	
<code>int getColumnType(int column)</code>	3.0	Yes	
<code>String getColumnTypeName(int column)</code>	3.0	Yes	
<code>int getPrecision(int column)</code>	3.0	Yes	
<code>int getScale(int column)</code>	3.0	Yes	
<code>String getSchemaName(int column)</code>	3.0	Yes	
<code>String getTableName(int column)</code>	3.0	Yes	
<code>boolean isAutoIncrement(int column)</code>	3.0	Yes	
<code>boolean isCaseSensitive(int column)</code>	3.0	Yes	
<code>boolean isCurrency(int column)</code>	3.0	Yes	
<code>boolean isDefinitelyWritable(int column)</code>	3.0	Yes	
<code>int isNullable(int column)</code>	3.0	Yes	
<code>boolean isReadOnly(int column)</code>	3.0	Yes	
<code>boolean isSearchable(int column)</code>	3.0	Yes	
<code>boolean isSigned(int column)</code>	3.0	Yes	
<code>boolean isWritable(int column)</code>	3.0	Yes	
<code>boolean isWrapperFor(Class<?> iface)</code>	4.0	Yes	
<code><T> T unwrap(Class<T> iface)</code>	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 Impala 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
<code>void addBatch(String sql)</code>	3.0	Yes	
<code>void cancel()</code>	3.0	Yes	
<code>void clearBatch()</code>	3.0	Yes	
<code>void clearWarnings()</code>	3.0	Yes	
<code>void close()</code>	3.0	Yes	
<code>void closeOnCompletion()</code>	4.1	Yes	
<code>boolean execute(String sql)</code>	3.0	Yes	
<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	
<code>int getFetchSize()</code>	3.0	Yes	
<code>ResultSet getGeneratedKeys()</code>	3.0	Yes	

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

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

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.

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.

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

Default Value	Data Type	Required
10	Integer	No

Auth_AccessToken

The OAuth 2.0 access token used for connecting to a server.

Default Value	Data Type	Required
None	String	Yes, if AuthMech=11.

Auth_Flow

This option specifies the type of OAuth authentication flow that the connector uses when the AuthMech property is set to 11.

- When this option is set to Token Passthrough (0), the connector uses the access token specified by (Auth_AccessToken) option to authenticate the connection to the server. For more information, see Auth_AccessToken.
- When this option is set to Client Credentials (1), the connector uses the client credentials to authenticate the connection to the server.
- When this option is set to Authorization Code (2), the connector uses the authorization code flow to authenticate the connection to the server.

Default Value	Data Type	Required
0 (Token passthrough)	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.
- 11 for OAuth 2.0.
- 12 for Single Sign-On.
- 14 for JWT.

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

AutoReconnect

This option specifies whether the connector attempts to automatically reconnect to the server and open a new session after receiving an invalid session error from the server.

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

Default Value	Data Type	Required
1	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 Impala server.

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

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 Impala catalogs as schemas or as catalogs.

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

Default Value	Data Type	Required
0	Integer	No

DataConversionInFetchThread

This property specifies whether to enable data conversion to be done in the fetch thread instead of the main thread for performance optimization.

- 1: The connector performs data conversion in the fetch thread which improves performance.
- 0: The connector does not perform data conversion in the fetch thread.

Default Value	Data Type	Required
1	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 Impala does not specify a maximum data length for STRING columns.

Default Value	Data Type	Required
255	Integer	No

DelegationUID

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

Default Value	Data Type	Required
None	String	No

EnableMultiSocketFetch

This properties specifies whether to enable the multi-socket fetch feature for the connection.

- 1: The connector enable multi-socket fetch.
- 0: The connector disables multi-socket fetch.

The multi-socket fetch feature is not supported when using Single Sign-On with SAML SSO (AuthMech=12).

Default Value	Data Type	Required
0	Boolean	No

httpPath

The partial URL corresponding to the Impala 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:impala://localhost:10002;AuthMech=3;transportMode=http;
httpPath=cliservice;UID=jsmith;PWD=cloudera123;
```

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

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

JWTString

The JSON Web Token used to access the server.

Default Value	Data Type	Required
None	String	Yes, if AuthMech=14 (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.
- 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 Impala host.

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

KrbRealm

The realm of the Impala host.

If your Kerberos configuration already defines the realm of the Impala 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 Impala server.

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

LogLevel

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

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 Impala, 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:

- An `ImpalaJDBC_driver.log` file that logs connector activity that is not specific to a connection.
- An `Impala_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.

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

LowerCaseResultSetColumnName

This property specifies the letter case that the connector uses when returning the column name aliases in the `ResultSetMetadata`.

- 1: The column name aliases in the `ResultSetMetadata` are returned in lower-case characters, matching the server-side behavior.
- 0: The column name aliases are returned in the same letter case as specified in the query.

Default Value	Data Type	Required
1	Integer	No

MaxNumMultiSocketFetchHandlers

The maximum number of threads to use to fetch multiple row sets concurrently when using the multi-socket fetch feature.

Default Value	Data Type	Required
8	Integer	No

MultiSocketFetchRowsetPerOperation

The maximum number of row sets to cache in memory for a query or an operation.

Default Value	Data Type	Required
10	Integer	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

OptimizedInsert

This property specifies whether the connector tries to optimize INSERT statements by bypassing translation.

Each time the connector translates an INSERT statement, it executes the DESCRIBE command to identify the data types of the columns that it is inserting data into. These additional commands consume resources and might reduce connector performance.

- 1: The connector tries to optimize INSERT statements by bypassing translation and using other methods to identify column types.
- 0: The connector does not attempt the optimization, and translates INSERT statements normally.

If the optimization fails, the connector falls back to translating INSERT statements normally. This additional overhead might further reduce connector performance.

Default Value	Data Type	Required
1	Integer	No

PreparedMetaLimitZero

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

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

Default Value	Data Type	Required
1	Integer	No

PWD

The password corresponding to the user name that you provided using the property `UID`.

Default Value	Data Type	Required
None	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

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 Impala 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.

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.

The Cloudera JDBC Connector for Apache Impala accepts TrustStores and KeyStores for one-way SSL authentication. See also the property SSLTrustStore.

Default Value	Data Type	Required
None	String	No

SSLKeyStoreProvider

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

Default Value	Data Type	Required
None	String	No

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

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

Default Value	Data Type	Required
JKS	String	No

SSLTrustStoreProvider

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

Default Value	Data Type	Required
None	String	No

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.

The Cloudera JDBC Connector for Apache Impala 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

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

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

Default Value	Data Type	Required
JKS	String	No

SSOWebServerTimeout

This property specifies the number of seconds that the connector waits before timing out while waiting for a browser response when authenticating using Single Sign-On (SSO).

If this property is set to 0, the connector will wait for an indefinite amount of time

Default Value	Data Type	Required
120	Integer	No

StripCatalogName

This property specifies whether the connector removes catalog names from query statements, if translation fails .

- 1: If query translation fails, then the connector removes catalog names from the query statement.
- 0: The connector does not remove catalog names from query statements.

Default Value	Data Type	Required
1	Integer	No

SupportTimeOnlyTimestamp

This property specifies whether the connector supports TIMESTAMP data that only contains a time value.

- 1: The connector supports TIMESTAMP data that only contains a time value.
- 0: The connector returns an error when working with TIMESTAMP data that only contains a time value.

Default Value	Data Type	Required
1	Integer	No

TransportMode

The transport protocol to use in the Thrift layer.

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

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=0` by default. This setting is valid only when the `AuthMech` property is set to 0, 3, or 12.

This option replaces and supersedes the deprecated `UseSasl` option.

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

UID

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

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

UpperCaseResultSetColName

This property specifies whether the connector converts the result set column name to upper case if translation fails or if the `UseNativeQuery` property is set to 1.

- `true`: If query translation fails or if the `UseNativeQuery` property is set to 1, the connector converts the result set column names to upper-case characters.
- `false`: The connector does not convert the result set column names to upper-case characters.

Default Value	Data Type	Required
<code>false</code>	Boolean	No

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 Impala SQL.
- 1: The connector does not transform the queries emitted by applications, so the native query is used.

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

Default Value	Data Type	Required
0	Integer	No

UseSasl (deprecated)

This option is deprecated. Use `TransportMode` instead (see `TransportMode`).

This property indicates if SASL is used in conjunction with the User Name and Password Authentication Mechanism (`AuthMech=3`).

- 0: No SASL authentication is used. User credentials are still passed to the server for services such as Sentry.
- 1: SASL authentication is used.

Default Value	Data Type	Required
1	Integer	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.

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.