cloudera
Ask Bigger Questions

Cloudera JDBC Driver for Impala
Version 2.5.5

Important Notice

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: 2.5.5

Date: September 9, 2014

Table of Contents

Introduction

Welcome to the Cloudera JDBC Driver for Impala. JDBC is one of the most established and widely supported APIs for connecting to and working with databases. At the heart of the technology is the JDBC driver, which connects an application to the database.

Cloudera JDBC Driver for 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 driver efficiently transforms an application's SQL query into the equivalent form in Impala SQL. Impala SQL is a subset of SQL-92. If an application is Impala-aware, then the driver is configurable to pass the query through. The driver 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, refer to the section *Features* on page 8.

Cloudera JDBC Driver for Impala complies with the JDBC 3.0 and JDBC 4.0 data standards.

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

System Requirements

To use the Cloudera JDBC Driver for Impala with the JDBC 3.0 API, each computer where you use the driver must meet the following system requirements:

- Java Runtime Environment (JRE) version 1.4 or 5.0

To use the Cloudera JDBC Driver for Impala with the JDBC 4.0 API, each computer where you use the driver must meet the following minimum system requirements:

- Java Runtime Environment (JRE) version 6.0

The driver is suitable for use with all versions of Cloudera Impala.

Cloudera JDBC Driver for Impala Files

The Cloudera JDBC Driver for Impala is delivered in two ZIP archives named **Cloudera_ImpalaJDBC3_<*version*>.zip** and **Cloudera_ImpalaJDBC4_<*version*>.zip**. Each archive contains the driver supporting the JDBC API version indicated in the archive name.

The **Cloudera_ImpalaJDBC3_<*version*>.zip** archive contains the following files and folder structure:

- ImpalaJDBC3
    - hive_metastore.jar

  - hive_service.jar
  - ImpalaJDBC3.jar
  - libfb303-0.9.0.jar
  - libthrift-0.9.0.jar
  - log4j-1.2.14.jar
  - ql.jar
  - slf4j-api-1.5.8.jar
  - slf4j-log4j12-1.5.8.jar
  - TCLIServiceClient.jar

The **Cloudera_ImpalaJDBC4_*<version>*.zip** archive contains the following files and folder structure:

- ImpalaJDBC4
  - hive_metastore.jar
  - hive_service.jar
  - ImpalaJDBC4.jar
  - libfb303-0.9.0.jar
  - libthrift-0.9.0.jar
  - log4j-1.2.14.jar
  - ql.jar
  - slf4j-api-1.5.8.jar
  - slf4j-log4j12-1.5.8.jar
  - TCLIServiceClient.jar

Using the Cloudera JDBC Driver for Impala

To access an Impala data warehouse using the Cloudera JDBC Driver for Impala, you must set:

- Class path
- Driver class
- Connection URL

The Cloudera JDBC Driver for Impala provides read-only access to Impala data.

Setting the Class Path

The class path is the path that the Java Runtime Environment searches for classes and other resource files. For details on setting the class path, refer to http://docs.oracle.com/javase/7/docs/technotes/tools/windows/classpath.html.

To use the Cloudera JDBC Driver for Impala, you must include all JAR files in the ZIP archive containing the driver you are using in the class path.

Cloudera JDBC Driver for Impala Classes

The Cloudera JDBC Driver for Impala makes a class with the following fully-qualified class name (FQCN) available, supporting JDBC 3.0 in connections to Impala instances:

- com.cloudera.impala.jdbc3.Driver

A class with the following FQCN is available supporting JDBC 4.0:

- com.cloudera.impala.jdbc4.Driver

Prior to connecting to the Impala server, initialize the class appropriate for the Impala server instance and your application as needed.

Building the Connection URL

Using the connection URL, you supply connection information to the data source that you are accessing. The connection URL for the Cloudera JDBC Driver for Impala takes the following form:

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

Placeholders in the connection URL are defined as follows:

- *Subprotocol* is the value **impala** in order to connect to an Impala system.
- *Host* is the DNS or IP address of the server hosting the Impala data warehouse.
- *Port* is the port to connect to on *Host*
- *Schema* is the name of the schema/database you want to access. Specifying a schema is optional.
- You can specify one or more connection properties. For details on all available properties, see *Appendix B: Driver Configuration Options* on page 11.

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

For example, to connect to an Impala instance installed on the local computer by using a user name and password:

```
jdbc:impala://localhost:21050;AuthMech=3;UID=UserName;PWD=Password
```

UserName and *Password* specify credentials for an existing user on the host running Impala.

Note: The UID and PWD properties apply when employing User Name and Password or User Name and Password with Secure Sockets Layer (SSL) authentication mechanisms. The UID property also applies when User Name authentication is enabled. For further details on properties available in the connection URL, see *Appendix B: Driver Configuration Options* on page 11.

Java Sample Code

The following Java code provides an example demonstrating how to use the JDBC API to:

- Register the Cloudera JDBC Driver for Impala
- Establish a connection to an Impala database
- Query the database
- Parse a result set
- Handle exceptions
- Clean up to avoid memory leakage

Important: To use the Cloudera JDBC Driver for Impala in an application, you must include all JAR files in the ZIP archive containing the driver you are using in the class path for your Java project.

```
// java.sql packages are required
import java.sql.*;
// java.math packages are required
import java.math.*;

class ClouderaJDBCImpalaExample {

   // Define a string as the fully qualified class name (FQCN) of
   // the desired JDBC driver
   static String JDBCDriver = "com.cloudera.impala.jdbc3.Driver";
   // Define a string as the connection URL
   static String ConnectionURL = "jdbc:impala://192.168.1.1:21050";

   public static void main(String[] args) {

        Connection con = null;
        Statement stmt = null;
        ResultSet rs = null;

        // Define the SQL statement for a query
        String query = "SELECT store_sales, store_cost FROM
             default.test LIMIT 20";

        try {

             // Register the driver using the class name
             Class.forName(JDBCDriver);

             // Establish a connection using the connection URL
             con = DriverManager.getConnection(ConnectionURL);
```

```
            // Create a Statement object for sending SQL
            // statements to the database
            stmt = con.createStatement();

            // Execute the SQL statement
            rs = stmt.executeQuery(query);

            // Display a header line for output appearing in
            // the Console View
            System.out.printf("%20s%20s\r\n", "STORE SALES",
                "STORE COST");

            // Step through each row in the result set returned
            // from the database
            while(rs.next()) {
                // Retrieve values from the row where the
                // cursor is currently positioned using column
                // names
                BigDecimal StoreSales = rs.getBigDecimal
                    ("store_sales");
                BigDecimal StoreCost = rs.getBigDecimal
                    ("store_cost");

                // Display values in columns 20 characters
                // wide in the Console View using the
                // Formatter
            System.out.printf("%20s%20s\r\n",
                StoreSales.toString(), StoreCost.toString());
            }

        } catch (SQLException se) {
            // Handle errors encountered during interaction
            // with the data source
            se.printStackTrace();
        } catch (Exception e) {
            // Handle other errors
            e.printStackTrace();
        } finally {
            // Perform clean up
            try {
                if (rs != null) {
                    rs.close();
                }
            } catch (SQLException se1) {
                  // Log this
             }

             try {
                  if (stmt!=null) {
                       stmt.close();
                  }
             } catch (SQLException se2) {
                  // Log this
             }

             try {
                  if (con!=null) {
                       con.close();
                  }
             } catch (SQLException se3) {
                  // Log this
                  se3.printStackTrace();
             } // End try

        } // End try

   } // End main
} // End ClouderaJDBCImpalaExample
```

Configuring Authentication

Impala supports the following authentication mechanisms:

- No Authentication
- Kerberos
- User Name
- User Name and Password
- User Name and Password with Secure Sockets Layer

For details on configuring the authentication mechanism that Impala uses, refer to the Impala documentation on the Cloudera web site .

When using the Cloudera JDBC Driver for Impala, the connection to Impala is authenticated using properties specified in the connection URL.

The following examples demonstrate how to connect using supported authentication mechanisms.

For details on properties you can specify in the connection URL, see *Appendix B: Driver Configuration Options* on page 11.

Using No Authentication

To use no authentication:

- ➢ Set the AuthMech property to 0

For example:

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

Using Kerberos

For information on operating Kerberos, refer to the documentation for your operating system.

To configure the Cloudera JDBC Driver for Impala to use Kerberos authentication:

1. Set the **AuthMech** property to 1.
2. 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 appropriate realm using the KrbRealm property.
3. Set the **KrbHostFQDN** property to the fully qualified domain name of the Impala host.
4. Set the **KrbServiceName** property to the service name of the Impala server.

For example:

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

Using User Name

To configure User Name authentication:

1. Set the **AuthMech** property to 2.
2. Set the **UID** property to the appropriate user name recognized by the Impala server.

For example:

```
jdbc:impala://localhost:21050;AuthMech=2;UID=impala
```

Using User Name and Password

To configure User Name and Password authentication:

1. Set the **AuthMech** property to 3.
2. Set the **UID** property to the appropriate user name recognized by the Impala server.
3. Set the **PWD** property to the password corresponding to the user name you provided in step 2.

For example:

```
jdbc:impala://localhost:21050;AuthMech=3;UID=impala;PWD=*****
```

Using User Name and Password with Secure Sockets Layer

To configure User Name and Password authentication using SSL:

1. Create a KeyStore containing your signed, trusted SSL certificate.
2. Set the **AuthMech** property to 4.
3. Set the **SSLKeyStore** property to the full path of the KeyStore you created in step 1, including the file name.
4. Set the **SSLKeyStorePwd** property to the password for the KeyStore you created in step 1.
5. Set the **UID** property to the appropriate user name recognized by the Impala server.
6. Set the **PWD** property to the password corresponding to the user name you provided in step 5.

For example:

```
jdbc:impala://localhost:21050;AuthMech=4;SSLKeyStore=C:\\Users\\b
smith\\Desktop\\keystore.jks;SSLKeyStorePwd=*****;UID=impala;PWD=
*****
```

Note: By default, the SSL certificate used by the server cannot be self-signed. Use the AllowSelfSignedCerts property to allow the SSL certificate used by the server to be self-signed. By default, the Common Name (CN) contained in a certificate signed by a Certification Authority (CA) must match the host name of the Impala server. You can allow mismatches using the CAIssuedCertNamesMismatch property. The driver always allows the Common Name contained in self-signed certificates to mismatch the host name. See *Appendix B: Driver Configuration Options* on page 11 for more information.

Features

Data Types

The following data types are supported:

- TINYINT
- SMALLINT
- INT
- BIGINT
- FLOAT
- DOUBLE

  Note: REAL is an alias for DOUBLE.

- BOOLEAN
- STRING
- TIMESTAMP

Catalog and Schema Support

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

Note: "CatalogSchemaSwitch" being set to 1 on the connection string will cause Impala Catalogs to be treated as Schemas in the Driver as a restriction for filtering.

SQL Translation

The Cloudera JDBC Driver for Impala is able to parse queries locally prior to sending them to the Impala server. This feature allows the driver 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.

Appendix A: Kerberos Authentication for Windows Configuration

Note: If you do not use Kerberos authentication, then you do not need to configure Kerberos authentication.

On the Windows platform, using Kerberos authentication as part of an Active Directory implementation is recommended. To implement Active Directory, contact your system administrator.

When using Kerberos for enhanced authentication in an Active Directory implementation, users are granted a Ticket to Get Tickets (TGT) automatically when logging on to the network. The Kerberos Key Distribution Center (KDC) is installed as part of the domain controller and performs two service functions: the Authentication Service (AS) and the Ticket-Granting Service (TGS). The KDC has access to Active Directory user account information. After the Authentication Service verifies the user, the Ticket-Granting Service provides the Ticket to Get Tickets.

In an Active Directory implementation using Kerberos, Kerberos uses the Active Directory cache to store credentials.

The Cloudera JDBC Driver for Impala requests a service ticket in the process of establishing a connection to the Impala database.

Appendix B: Driver Configuration Options

Table 1 lists and describes the properties available to configure the behavior of the Cloudera JDBC Driver for Impala.

Note: You can set configuration properties using the connection URL. For details on the connection URL, see *Building the Connection URL* on page 3.

Key	Default Value	Description
AuthMech	0	The authentication mechanism to use. Set the value to one of the following: • 0 for no authentication • 1 for Kerberos • 2 for User Name • 3 for User Name and Password • 5 for User Name and Password (SSL)
DefaultStringColumnLength	255	The maximum data length for string columns. **Note:** Impala does not provide the maximum data length for String columns in the columns metadata. This option allows you to tune the maximum data length for String columns. (Optional)
DelegationUID		Used to delegate all operations against Impala to a user that is different than the authenticated user for the connection.
KrbFQDN		The fully qualified domain name of the Impala host used.
KrbRealm		If there is no default realm configured or the realm of the Impala host is different from the default realm for your Kerberos setup, then define the realm of the Impala host using this option.
KrbServiceName		The Kerberos service principal name of the Impala server.

Key	Default Value	Description
PWD		The password of a user account on the host running Impala. PWD is required if AuthMech is set to User Name and Password, or User Name and Password (SSL).
RowsFetchedPerBlock	10000	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.
SocketTimeout	0	The number of seconds after which Impala closes the connection with the client application if the connection is idle. The default value of 0 indicates that an idle connection is not closed.
SSLKeyStore	N/A	The full path, including the file name, of the Java KeyStore containing an SSL certificate to use during authentication. See also the SSLKeyStorePwd property. SSLKeyStore is required if AuthMech is User Name and Password (SSL).
SSLKeyStorePwd	N/A	The password required to access the Java KeyStore specified using the SSLKeyStore property. SSLKeyStorePwd is required if AuthMech is User Name and Password (SSL).
UID	anonymous	The user name of an existing account on the host running Impala. UID is required if AuthMech is set to User Name and Password, or User Name and Password (SSL). UID is optional if AuthMech is set to User Name.

Key	Default Value	Description
UseNativeQuery	0	Enabling the UseNativeQuery option using a value of 1 disables the feature of the driver to apply transformations to the queries emitted by an application to convert the queries into an equivalent form in Impala SQL. If the application is Impala aware and already emits Impala SQL, then disabling the feature avoids the extra overhead of query transformation.
CatalogSchemaSwitch	0	Enabling the CatalogSchemaSwitch option using a value of 1 on the connection string will cause Impala Catalogs to be treated as Schemas in the Driver as a restriction for filtering.
PreparedMetaLimitZero	1	Enabling PreparedMetaLimitZero will cause the PreparedStatement.getMetadata() call to request metadata from the server with “LIMIT 0”, increasing performance.

Table 1 Driver Configuration Options