



TIBCO® Data Virtualization

Cassandra Adapter Guide

Version 8.7.0 | October 2023

Contents

Contents	2
Cassandra Adapter	4
Getting Started	4
Basic Tab	5
Logging	7
Using Kerberos	9
Fine-Tuning Data Access	10
NoSQL Database	11
Automatic Schema Discovery	11
Data Type Mapping	13
Changelog	15
Advanced Features	17
User Defined Views	18
SSL Configuration	21
Firewall and Proxy	22
Query Processing	22
Logging	23
SQL Compliance	26
SELECT Statements	27
SELECT INTO Statements	30
INSERT Statements	31
UPDATE Statements	32
DELETE Statements	32
EXECUTE Statements	33
PIVOT and UNPIVOT	34
Connection String Options	35
Authentication	40

Kerberos	47
SSL	52
SSH	58
Firewall	66
Logging	69
Schema	70
Miscellaneous	71
TIBCO Product Documentation and Support Services	85
How to Access TIBCO Documentation	85
How to Contact TIBCO Support	86
Release Version Support	86
How to Join TIBCO Community	87
Legal and Third-Party Notices	88

Cassandra Adapter

Cassandra Version Support

The adapter supports CQL versions 2.0, 3.0 and 4.0

SQL Compliance

The [SQL Compliance](#) section shows the SQL syntax supported by the adapter and points out any limitations.

Getting Started

Connecting to Cassandra

[Basic Tab](#) shows how to authenticate to Cassandra and configure any necessary connection properties. Additional adapter capabilities can be configured using the available [Connection](#) properties on the Advanced tab. The Advanced Settings section shows how to set up more advanced configurations and troubleshoot connection errors.

Deploying the Cassandra Adapter

To deploy the adapter, you can execute the `server_util` utility via the command line by

1. Unzip the `tdv.cassandra.zip` file to the location of your choice.
2. Open a command prompt window.
3. Navigate to the `<TDV_install_dir>/bin`
4. Enter the `server_util` command with the `-deploy` option:

```
server_util -server <hostname> [-port <port>] -user <user> -  
password <password> -deploy -package <TDV_install_  
dir>/adapters/tdv.cassandra/tdv.cassandra.jar
```

Note: When deploying a build of an existing adapter, you will need to undeploy the existing adapter using the `server_util` command with the `-undeploy` option.

```
server_util -server <hostname> [-port <port>] -user <user> -password
<password> -undeploy -version 1 -name Cassandra
```

Basic Tab

Connecting to Cassandra

Log in to the Azure Portal, select **Azure Cosmos DB**, and select your account. In the **Settings** section, click **Connection String** and set the following values:

- Server: Set this to the Host value, the FQDN of the server provisioned for your account. You can also specify the port here or in Port.
- Port: Set this to the port on which the Cassandra database is hosted.
- Database: Set this to the database you want to read from and write to.
- ConsistencyLevel: Set this to the number of the replicas that you want to enforce a response from before queries are considered a success.
- User: Set this to the Cosmos DB account name.
- Password: The account key associated with the Cosmos DB account.

Authenticating to Cassandra

The adapter supports Basic authentication with login credentials and the additional authentication features of DataStax Enterprise (DSE). The following sections detail connection properties your authentication method may require.

You need to set AuthScheme to the value corresponding to the authenticator configured for your system. You specify the authenticator in the *authenticator* property in the **cassandra.yaml** file. This file is typically found in */etc/dse/cassandra* or through the **DSE Unified Authenticator** on DSE Cassandra.

Basic

Set AuthScheme to **Basic** to authenticate with login credentials alone.

In the *cassandra.yaml* file, set the *authenticator* property to "PasswordAuthenticator".

DSE

Set the AuthScheme property to **DSE** to authenticate with login credentials and the DSE Unified Authenticator.

In the file, set the *authenticator* property to "com.datastax.bdp.cassandra.auth.DseAuthenticator".

Kerberos

Set the following to authenticating using Kerberos:

- AuthScheme: Set this to **KERBEROS**.
- KerberosKDC: Set this to the Kerberos Key Distribution Center (KDC) service used to authenticate the user.
- KerberosRealm: Set this to the Kerberos Realm used to authenticate the user.
- KerberosSPN: Set this to the service principal name (SPN) for the Kerberos Domain Controller.

Next, configure these YAML files as described below:

- In the `cassandra.yaml` file, set the *authenticator* property to "com.datastax.bdp.cassandra.auth.DseAuthenticator".
- Modify the **authentication_options** section in the `dse.yaml` file, specifying the *default_schema* and *other_schemas* properties as "kerberos".
- Modify the **kerberos_options** section in the `dse.yaml` file, specifying the *keytab*, *service_principle*, *http_principle* and *qop* properties.

Please see [Using Kerberos](#) for more details on how to set connection properties in order to connect to Kerberos.

LDAP

Set the following to authenticating using Kerberos:

- AuthScheme: Set this to **LDAP** to authenticate an LDAP user.
- LDAPServer: Set this to the host name or IP address of the LDAP server.

- LDAPPassword: The password of the default LDAP user.

Next, configure these YAML files as described below:

- In the `cassandra.yaml` file, set the `authenticator` property to `"com.datastax.bdp.cassandra.auth.DseAuthenticator"`.
- Modify the `authentication_options` section in the `dse.yaml` file, specifying the `default_schema` and `other_schemas` properties as `"ldap"`.
- Modify the `ldap_options` section in the `dse.yaml` file, specifying the `server_host`, `server_port`, `search_dn`, `search_password`, `user_search_base`, and `user_search_filter` properties.

Securing Cassandra Connections

You can set UseSSL to negotiate SSL/TLS encryption when you connect. By default, the adapter attempts to negotiate SSL/TLS by checking the server's certificate against the system's trusted certificate store. To specify another certificate, see the SSLServerCert property for the available formats.

Logging

The adapter uses TDV Server's logging (log4j) to generate log files. The settings within the TDV Server's logging (log4j) configuration file are used by the adapter to determine the type of messages to log. The following categories can be specified:

- Error: Only error messages are logged.
- Info: Both Error and Info messages are logged.
- Debug: Error, Info, and Debug messages are logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, that is:

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

- 0 = Error
- 1-2 = Info

- 3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter never logs at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity defaults to the minimum value for that particular category. For example, if Verbosity is set to a value less than 3 and the Debug category is specified, the Verbosity defaults to 3.

The following list is an explanation of the Verbosity levels and the information that they log.

- 1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.
- 2 - Will log everything included in Verbosity 1 and HTTP headers.
- 3 - Will additionally log the body of the HTTP requests.
- 4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.
- 5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Cassandra Adapter

By default, logging is turned on without debugging. If debugging information is desired, uncomment the following line in the TDV Server's log4j.properties file (default location of this file is: C:\Program Files\TIBCO\TDV Server <version>\conf\server):

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server must be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at: C:\Program Files\TIBCO\TDV Server <version>\bin. Note that reauthenticating to the TDV Studio is required after restarting the server.

Here is an example of the calls:

```
.\composite.bat monitor restart
```


All logs for the adapter are written to the "cs_server_dsrc.log" file as specified in the log4j properties.

Note: The "log4j.logger.com.cdata=DEBUG" option is not required if the **Debug Output Enabled** option is set to true within the TDV Studio. To set this option, navigate to **Administrator > Configuration**. Select **Server > Configuration > Debugging** and set the Debug Output Enabled option to **True**.

Using Kerberos

This section shows how to use the adapter to authenticate using Kerberos.

Kerberos

To authenticate to Cassandra using Kerberos, set the following properties:

- AuthScheme: Set this to **KERBEROS**.
- KerberosKDC: Set this to the host name or IP Address of your Kerberos KDC machine.
- KerberosRealm: Set this to **the realm of the Cassandra Kerberos principal**. This will be the value after the '@' symbol (for instance, EXAMPLE.COM) of the **principal value** (for instance, hbase/MyHost@EXAMPLE.COM).
- KerberosSPN: Set this to the service and host of the Cassandra Kerberos Principal. This is the value prior to the '@' symbol (for instance, hbase/MyHost) of the principal value (for instance, hbase/MyHost@EXAMPLE.COM).

Retrieve the Kerberos Ticket

You can use one of the following options to retrieve the required Kerberos ticket.

MIT Kerberos Credential Cache File

This option enables you to use the MIT Kerberos Ticket Manager or kinit command to get tickets. Note that you do not need to set the User or Password connection properties with this option.

1. Ensure that you have an environment variable created called **KRB5CCNAME**.
2. Set the **KRB5CCNAME** environment variable to a path pointing to your credential cache file (for instance, C:\krb_cache\krb5cc_0 or /tmp/krb5cc_0). This file is created

when generating your ticket with MIT Kerberos Ticket Manager.

3. To obtain a ticket, open the MIT Kerberos Ticket Manager application, click **Get Ticket**, enter your principal name and password, then click **OK**. If successful, ticket information appears in Kerberos Ticket Manager and is stored in the credential cache file.
4. Now that you have created the credential cache file, the adapter uses the cache file to obtain the Kerberos ticket to connect to Cassandra.

As an alternative to setting the **KRB5CCNAME** environment variable, you can directly set the file path using the KerberosTicketCache property. When set, the adapter uses the specified cache file to obtain the Kerberos ticket to connect to Cassandra.

Keytab File

If the **KRB5CCNAME environment variable has not been set**, you can retrieve a Kerberos ticket using a Keytab File. To do so, set the User property to the desired username and set the KerberosKeytabFile property to a file path pointing to the keytab file associated with the user.

User and Password

If both the **KRB5CCNAME** environment variable and the KerberosKeytabFile property have not been set, you can retrieve a ticket using a user and password combination. To do this, set the User and Password properties to the user/password combination that you use to authenticate with Cassandra.

Cross-Realm

More complex Kerberos environments may require cross-realm authentication where multiple realms and KDC servers are used (e.g., where one realm/KDC is used for user authentication and another realm/KDC is used for obtaining the service ticket).

In such an environment, set the KerberosRealm and KerberosKDC properties to the values required for user authentication. Also set the KerberosServiceRealm and KerberosServiceKDC properties to the values required to obtain the service ticket.

Fine-Tuning Data Access

Fine Tuning Data Access

You can use the following properties to gain greater control over Cassandra API features and the strategies the adapter uses to surface them:

- AllowFiltering: Set this property to allow the server to process slow-performing searches.
- UseJsonFormat: Set this property to use CQL literals instead of JSON.
- QueryPassthrough: This property enables you to use native CQL statements instead of SQL.
- RowScanDepth: This property determines the number of rows that will be scanned to detect column data types when generating table metadata.

This property applies if you are working with the dynamic schemas generated from [Automatic Schema Discovery](#) or if you are using QueryPassthrough.

NoSQL Database

Cassandra is a NoSQL database that provides high performance, availability, and scalability. However, these capabilities are not necessarily incompatible with a standards-compliant query language like SQL-92. The adapter models Cassandra tables into relational tables and translates SQL queries into calls to the Cassandra API, the CQL (Cassandra Query Language) binary protocol.

The equivalent of a table in Cassandra is a column family. Column families contain columns of related data. Like other NoSQL databases, Cassandra allows complex types of fields such as set, list, and map. A column family is a nested map data structure. This can be represented as a JSON object.

The adapter offers two ways to model Cassandra objects. The [Automatic Schema Discovery](#) scheme automatically finds the data types in a Cassandra object by scanning a configured number of rows of the object. You can use RowScanDepth, FlattenArrays, and FlattenObjects to control the relational representation of the tables in Cassandra.

Automatic Schema Discovery

The adapter automatically infers a relational schema by inspecting a series of Cassandra documents in a collection. You can use the RowScanDepth property to define the number

of documents the adapter will scan to do so. The columns identified during the discovery process depend on the [FlattenArrays](#) and [FlattenObjects](#) properties.

Flattening Objects

If [FlattenObjects](#) is set, all nested objects will be flattened into a series of columns. For example, consider the following document:

```
{
  id: 12,
  name: "Lohia Manufacturers Inc.",
  address: {street: "Main Street", city: "Chapel Hill", state: "NC"},
  offices: ["Chapel Hill", "London", "New York"],
  annual_revenue: 35,600,000
}
```

This document will be represented by the following columns:

Column Name	Data Type	Example Value
id	Integer	12
name	String	Lohia Manufacturers Inc.
address.street	String	Main Street
address.city	String	Chapel Hill
address.state	String	NC
offices	String	["Chapel Hill", "London", "New York"]
annual_revenue	Double	35,600,000

If [FlattenObjects](#) is not set, then the address.street, address.city, and address.state columns will not be broken apart. The address column of type string will instead represent the entire object. Its value would be `{street: "Main Street", city: "Chapel Hill", state: "NC"}`. See [JSON Functions](#) for more details on working with JSON aggregates.

Flattening Arrays

The `FlattenArrays` property can be used to flatten array values into columns of their own. This is only recommended for arrays that are expected to be short, for example the coordinates below:

```
"coord": [ -73.856077, 40.848447 ]
```

The `FlattenArrays` property can be set to 2 to represent the array above as follows:

Column Name	Data Type	Example Value
coord.0	Float	-73.856077
coord.1	Float	40.848447

It is best to leave other unbounded arrays as they are and piece out the data for them as needed using [JSON Functions](#).

Data Type Mapping

Data Type Mappings

The adapter maps types from the data source to the corresponding data type available in the schema. The table below documents these mappings.

Note that string columns can map to different data types depending on their length.

Cassandra	CData Schema
ascii	string
bigint	long
blob	binary

boolean	bool
counter	long
date	date
decimal	decimal
double	float
float	float
inet	string
int	int
list	string
map	string
set	string
smallint	int
text	string
time	time
timestamp	datetime
timeuuid	string
tinyint	int
tuple	string
udt	string
uuid	string

varchar	string
varint	string

Changelog

General Changes

Date	Build Number	Change Type	Description
12/14/2022	8383	General	Changed <ul style="list-style-type: none"> Added the Default column to the sys_procedureparameters table.
09/30/2022	8308	General	Changed <ul style="list-style-type: none"> Added the IsPath column to the sys_procedureparameters table.
08/17/2022	8264	General	Changed <ul style="list-style-type: none"> We now support handling the keyword "COLLATE" as standard function name as well.
09/02/2021	7915	General	Added <ul style="list-style-type: none"> Added support for the STRING_SPLIT table-valued function in the CROSS APPLY clause.
08/07/2021	7889	General	Changed <ul style="list-style-type: none"> Added the KeySeq column to the sys_foreignkeys table.

08/06/2021	7888	General	Changed <ul style="list-style-type: none"> Added the new sys_primarykeys system table.
07/23/2021	7874	General	Changed <ul style="list-style-type: none"> Updated the Literal Function Names for relative date/datetime functions. Previously relative date/datetime functions resolved to a different value when used in the projection vs te predicate. Ie: SELECT LAST_MONTH() AS lm, Col FROM Table WHERE Col > LAST_MONTH(). Formerly the two LAST_MONTH() methods would resolve to different datetimes. Now they will match. As a replacement for the previous behavior, the relative date/datetime functions in the criteria may have an 'L' appended to them. Ie: WHERE col > L_LAST_MONTH(). This will continue to resolve to the same values that previously were calculated in the criteria. Note that the "L_" prefix will only work in the predicate - it not available for the projection.
07/08/2021	7859	General	Added <ul style="list-style-type: none"> Added the TCP Logging Module for the logging information happening on the TCP wire protocol. The transport bytes that are incoming and ongoing will be logged at verbosity=5.
04/23/2021	7785	General	Added <ul style="list-style-type: none"> Added support for handling client side formulas during insert / update. For example: UPDATE Table SET Col1 = Concat (Col1, " - ", Col2) WHERE Col2 LIKE 'A%'

04/23/2021	7783	General	Changed <ul style="list-style-type: none"> Updated how display sizes are determined for varchar primary key and foreign key columns so they will match the reported length of the column.
04/16/2021	7776	General	Added <ul style="list-style-type: none"> Non-conditional updates between two columns is now available to all drivers. For example: UPDATE Table SET Col1=Col2 Changed <ul style="list-style-type: none"> Reduced the length to 255 for varchar primary key and foreign key columns. Updated implicit and metadata caching to improve performance and support for multiple connections. Old metadata caches are not compatible - you would need to generate new metadata caches if you are currently using CacheMetadata. Updated index naming convention to avoid duplicates Updated and standardized Getting Started connection help. Added the Advanced Features section to the help of all drivers. Categorized connection property listings in the help for all editions.
04/15 /2021	7775	General	Changed <ul style="list-style-type: none"> Kerberos authentication is updated to use TCP by default, but will fall back to UDP if a TCP connection cannot be established

Advanced Features

This section details a selection of advanced features of the Cassandra adapter.

User Defined Views

The adapter allows you to define virtual tables, called *user defined views*, whose contents are decided by a pre-configured query. These views are useful when you cannot directly control queries being issued to the drivers. See [User Defined Views](#) for an overview of creating and configuring custom views.

SSL Configuration

Use [SSL Configuration](#) to adjust how adapter handles TLS/SSL certificate negotiations. You can choose from various certificate formats; see the `SSLServerCert` property under "Connection String Options" for more information.

Firewall and Proxy

Configure the adapter for compliance with [Firewall and Proxy](#), including Windows proxies. You can also set up tunnel connections.

Query Processing

The adapter offloads as much of the SELECT statement processing as possible to Cassandra and then processes the rest of the query in memory (client-side).

See [Query Processing](#) for more information.

Logging

See [Logging](#) for an overview of configuration settings that can be used to refine CData logging. For basic logging, you only need to set two connection properties, but there are numerous features that support more refined logging, where you can select subsets of information to be logged using the `LogModules` connection property.

User Defined Views

The Cassandra Adapter allows you to define a virtual table whose contents are decided by a pre-configured query. These are called *User Defined Views*, which are useful in situations

where you cannot directly control the query being issued to the driver, e.g. when using the driver from a tool. The User Defined Views can be used to define predicates that are always applied. If you specify additional predicates in the query to the view, they are combined with the query already defined as part of the view.

There are two ways to create user defined views:

- Create a JSON-formatted configuration file defining the views you want.
- DDL statements.

Defining Views Using a Configuration File

User Defined Views are defined in a JSON-formatted configuration file called *UserDefinedViews.json*. The adapter automatically detects the views specified in this file.

You can also have multiple view definitions and control them using the UserDefinedViews connection property. When you use this property, only the specified views are seen by the adapter.

This User Defined View configuration file is formatted as follows:

- Each root element defines the name of a view.
- Each root element contains a child element, called **query**, which contains the custom SQL query for the view.

For example:

```
{
  "MyView": {
    "query": "SELECT * FROM \"CData\\\".\"Sample\\\".Products WHERE MyColumn =
'value'\"
  },
  "MyView2": {
    "query": "SELECT * FROM MyTable WHERE Id IN (1,2,3)"
  }
}
```

Use the UserDefinedViews connection property to specify the location of your JSON configuration file. For example:

```
"UserDefinedViews",
"C:\\Users\\yourusername\\Desktop\\tmp\\UserDefinedViews.json"
```

Defining Views Using DDL Statements

The adapter is also capable of creating and altering the schema via DDL Statements such as CREATE LOCAL VIEW, ALTER LOCAL VIEW, and DROP LOCAL VIEW.

Create a View

To create a new view using DDL statements, provide the view name and query as follows:

```
CREATE LOCAL VIEW [MyViewName] AS SELECT * FROM Customers LIMIT 20;
```

If no JSON file exists, the above code creates one. The view is then created in the JSON configuration file and is now discoverable. The JSON file location is specified by the UserDefinedViews connection property.

Alter a View

To alter an existing view, provide the name of an existing view alongside the new query you would like to use instead:

```
ALTER LOCAL VIEW [MyViewName] AS SELECT * FROM Customers WHERE  
TimeModified > '3/1/2020';
```

The view is then updated in the JSON configuration file.

Drop a View

To drop an existing view, provide the name of an existing schema alongside the new query you would like to use instead.

```
DROP LOCAL VIEW [MyViewName]
```

This removes the view from the JSON configuration file. It can no longer be queried.

Schema for User Defined Views

User Defined Views are exposed in the **UserViews** schema by default. This is done to avoid the view's name clashing with an actual entity in the data model. You can change the name of the schema used for UserViews by setting the UserViewsSchemaName property.

Working with User Defined Views

For example, a SQL statement with a User Defined View called *UserViews.RCustomers* only lists customers in Raleigh:

```
SELECT * FROM Customers WHERE City = 'Raleigh';
```

An example of a query to the driver:

```
SELECT * FROM UserViews.RCustomers WHERE Status = 'Active';
```

Resulting in the effective query to the source:

```
SELECT * FROM Customers WHERE City = 'Raleigh' AND Status = 'Active';
```

That is a very simple example of a query to a User Defined View that is effectively a combination of the view query and the view definition. It is possible to compose these queries in much more complex patterns. All SQL operations are allowed in both queries and are combined when appropriate.

SSL Configuration

Customizing the SSL Configuration

By default, the adapter attempts to negotiate SSL/TLS by checking the server's certificate against the system's trusted certificate store.

To specify another certificate, see the [SSLServerCert](#) property for the available formats to do so.

Client SSL Certificates

The Cassandra adapter also supports setting client certificates. Set the following to connect using a client certificate.

- [SSLClientCert](#): The name of the certificate store for the client certificate.
- [SSLClientCertType](#): The type of key store containing the TLS/SSL client certificate.
- [SSLClientCertPassword](#): The password for the TLS/SSL client certificate.
- [SSLClientCertSubject](#): The subject of the TLS/SSL client certificate.

Firewall and Proxy

Connecting Through a Firewall or Proxy

Set the following properties:

- To use a proxy-based firewall, set FirewallType, FirewallServer, and FirewallPort.
- To tunnel the connection, set FirewallType to TUNNEL.
- To authenticate, specify FirewallUser and FirewallPassword.
- To authenticate to a SOCKS proxy, additionally set FirewallType to SOCKS5.

Query Processing

Query Processing

CData has a client-side SQL engine built into the adapter library. This enables support for the full capabilities that SQL-92 offers, including filters, aggregations, functions, etc.

For sources that do not support SQL-92, the adapter offloads as much of SQL statement processing as possible to Cassandra and then processes the rest of the query in memory (client-side). This results in optimal performance.

For data sources with limited query capabilities, the adapter handles transformations of the SQL query to make it simpler for the adapter. The goal is to make smart decisions based on the query capabilities of the data source to push down as much of the computation as possible. The Cassandra Query Evaluation component examines SQL queries and returns information indicating what parts of the query the adapter is not capable of executing natively.

The Cassandra Query Slicer component is used in more specific cases to separate a single query into multiple independent queries. The client-side Query Engine makes decisions about simplifying queries, breaking queries into multiple queries, and pushing down or computing aggregations on the client-side while minimizing the size of the result set.

There's a significant trade-off in evaluating queries, even partially, client-side. There are always queries that are impossible to execute efficiently in this model, and some can be particularly expensive to compute in this manner. CData always pushes down as much of the query as is feasible for the data source to generate the most efficient query possible and provide the most flexible query capabilities.

More Information

For a full discussion of how CData handles query processing, see [CData Architecture: Query Execution](#).

Logging

Capturing adapter logging can be very helpful when diagnosing error messages or other unexpected behavior.

Basic Logging

You will simply need to set two connection properties to begin capturing adapter logging.

- Logfile: A filepath which designates the name and location of the log file.
- Verbosity: This is a numerical value (1-5) that determines the amount of detail in the log. See the page in the Connection Properties section for an explanation of the five levels.
- MaxLogFileSize: When the limit is hit, a new log is created in the same folder with the date and time appended to the end. The default limit is 100 MB. Values lower than 100 kB will use 100 kB as the value instead.
- MaxLogFileCount: A string specifying the maximum file count of log files. When the limit is hit, a new log is created in the same folder with the date and time appended to the end and the oldest log file will be deleted. Minimum supported value is 2. A value of 0 or a negative value indicates no limit on the count.

Once this property is set, the adapter will populate the log file as it carries out various tasks, such as when authentication is performed or queries are executed. If the specified file doesn't already exist, it will be created.

Log Verbosity

The verbosity level determines the amount of detail that the adapter reports to the Logfile. Verbosity levels from 1 to 5 are supported. These are described in the following list:

-
- | | |
|---|---|
| 1 | Setting <u>Verbosity</u> to 1 will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors. |
|---|---|
-

-
- | | |
|---|--|
| 2 | Setting <u>Verbosity</u> to 2 will log everything included in <u>Verbosity</u> 1 and additional information about the request. |
|---|--|
-
- | | |
|---|---|
| 3 | Setting <u>Verbosity</u> to 3 will additionally log the body of the request and the response. |
|---|---|
-
- | | |
|---|--|
| 4 | Setting <u>Verbosity</u> to 4 will additionally log transport-level communication with the data source. This includes SSL negotiation. |
|---|--|
-
- | | |
|---|--|
| 5 | Setting <u>Verbosity</u> to 5 will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands. |
|---|--|
-

The Verbosity should not be set to greater than 1 for normal operation. Substantial amounts of data can be logged at higher verbosity levels, which can delay execution times.

To refine the logged content further by showing/hiding specific categories of information, see LogModules.

Sensitive Data

Verbosity levels 3 and higher may capture information that you do not want shared outside of your organization. The following lists information of concern for each level:

- Verbosity 3: The full body of the request and the response, which includes all the data returned by the adapter
- Verbosity 4: SSL certificates
- Verbosity 5: Any extra transfer data not included at Verbosity 3, such as non human-readable binary transfer data

Best Practices for Data Security

Although we mask sensitive values, such as passwords, in the connection string and any request in the log, it is always best practice to review the logs for any sensitive information before sharing outside your organization.

Java Logging

When Java logging is enabled in Logfile, the Verbosity will instead map to the following logging levels.

- 0: Level.WARNING
- 1: Level.INFO
- 2: Level.CONFIG
- 3: Level.FINE
- 4: Level.FINER
- 5: Level.FINEST

Advanced Logging

You may want to refine the exact information that is recorded to the log file. This can be accomplished using the LogModules property.

This property allows you to filter the logging using a semicolon-separated list of logging modules.

All modules are four characters long. **Please note that modules containing three letters have a required trailing blank space.** The available modules are:

- **EXEC:** Query Execution. Includes execution messages for original SQL queries, parsed SQL queries, and normalized SQL queries. Query and page success/failure messages appear here as well.
- **INFO:** General Information. Includes the connection string, driver version (build number), and initial connection messages.
- **HTTP:** HTTP Protocol messages. Includes HTTP requests/responses (including POST messages), as well as Kerberos related messages.
- **SSL :** SSL certificate messages.
- **OAUT:** OAuth related failure/success messages.
- **SQL :** Includes SQL transactions, SQL bulk transfer messages, and SQL result set messages.
- **META:** Metadata cache and schema messages.
- **TCP :** Incoming and Ongoing raw bytes on TCP transport layer messages.

An example value for this property would be.

```
LogModules=INFO;EXEC;SSL ;SQL ;META;
```

Note that these modules refine the information as it is pulled after taking the Verbosity into account.

SQL Compliance

The Cassandra Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See [SELECT Statements](#) for a syntax reference and examples.

See [NoSQL Database](#) for information on the capabilities of the Cassandra API.

INSERT Statements

See [INSERT Statements](#) for a syntax reference and examples.

UPDATE Statements

The primary key Id is required to update a record. See [UPDATE Statements](#) for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See [DELETE Statements](#) for a syntax reference and examples.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See [EXECUTE Statements](#) for a syntax reference and examples.

Names and Quoting

- Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Z, a-z, 0-9, _:@].

- To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.
- Strings must be quoted using single quotes (e.g., 'John Doe').

SELECT Statements

A SELECT statement can consist of the following basic clauses.

- SELECT
- INTO
- FROM
- JOIN
- WHERE
- GROUP BY
- HAVING
- UNION
- ORDER BY
- LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Cassandra adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
      <expression> [ [ AS ] <column_reference> ]
      | { <table_name> | <correlation_name> } .*
    } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
```

```

[
  ORDER BY
    <column_reference> [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ]
]
[
  LIMIT <expression>
]
} | SCOPE_IDENTITY()
<expression> ::=
  | <column_reference>
  | @ <parameter>
  | ?
  | COUNT( * | { [ DISTINCT ] <expression> } )
  | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
  | NULLIF ( <expression> , <expression> )
  | COALESCE ( <expression> , ... )
  | CASE <expression>
    WHEN { <expression> | <search_condition> } THEN { <expression> |
NULL } [ ... ]
  [ ELSE { <expression> | NULL } ]
  END
  | <literal>
  | <sql_function>
<search_condition> ::=
  {
    <expression> { = | > | < | >= | <= | IN | AND } [ <expression> ]
  } [ { AND | OR } ... ]

```

Examples

1. Return all columns:

```
SELECT * FROM "CData"."Sample".Products
```

2. Rename a column:

```
SELECT "Name" AS MY_Name FROM "CData"."Sample".Products
```

3. Cast a column's data as a different data type:

```
SELECT CAST(AnnualRevenue AS VARCHAR) AS Str_AnnualRevenue FROM
```

```
"CData"."Sample".Products
```

4. Search data:

```
SELECT * FROM "CData"."Sample".Products WHERE Industry = 'Floppy  
Disks'
```

5. The Cassandra APIs support the following operators in the WHERE clause: =, >, <, >=, <=, IN, AND.

```
SELECT * FROM "CData"."Sample".Products WHERE Industry = 'Floppy  
Disks';
```

6. Return the number of items matching the query criteria:

```
SELECT COUNT(*) AS MyCount FROM "CData"."Sample".Products
```

7. Summarize data:

```
SELECT MAX(AnnualRevenue) FROM "CData"."Sample".Products
```

See [Aggregate Functions](#) for details.

8. Sort a result set in ascending order:

```
SELECT Id, Name FROM "CData"."Sample".Products ORDER BY Name ASC
```

Aggregate Functions

Examples of Aggregate Functions

Below are several examples of SQL aggregate functions.

COUNT

Returns the number of rows matching the query criteria.

```
SELECT COUNT(*) FROM "CData"."Sample".Products WHERE Industry = 'Floppy  
Disks'
```

AVG

Returns the average of the column values.

```
SELECT  AVG(AnnualRevenue) FROM "CData"."Sample".Products WHERE Industry  
= 'Floppy Disks'
```

MIN

Returns the minimum column value.

```
SELECT MIN(AnnualRevenue) FROM "CData"."Sample".Products WHERE Industry  
= 'Floppy Disks'
```

MAX

Returns the maximum column value.

```
SELECT  MAX(AnnualRevenue) FROM "CData"."Sample".Products WHERE Industry  
= 'Floppy Disks'
```

SUM

Returns the total sum of the column values.

```
SELECT SUM(AnnualRevenue) FROM "CData"."Sample".Products WHERE Industry  
= 'Floppy Disks'
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
boolean ret = stat.execute("SELECT Id, Name INTO
'csv://c:/\"CData\\\".\"Sample\\\".Products.txt\" FROM
\"CData\\\".\"Sample\\\".Products\" WHERE Industry = 'Floppy Disks'");
System.out.println(stat.getUpdateCount()+" rows affected");
```

You can specify other file formats in the URI. The following example exports tab-separated values:

```
Statement stat = conn.createStatement();
boolean ret = stat.execute("SELECT * INTO
\"CData\\\".\"Sample\\\".Products\" IN
'csv://filename=c:/\"CData\\\".\"Sample\\\".Products.csv;delimiter=tab' FROM
\"CData\\\".\"Sample\\\".Products\" WHERE Industry = 'Floppy Disks'");
System.out.println(stat.getUpdateCount()+" rows affected");
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause, as shown in the following example:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )

<expression> ::=
| @ <parameter>
| ?
| <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "INSERT INTO \"CData\".\"Sample\".Products (Name) VALUES
(?)" ;
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "John");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause, as shown in the following example:

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ , ... ]
WHERE { Id = <expression> } [ { AND | OR } ... ]
<expression> ::=
    | @ <parameter>
    | ?
    | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected, as shown in the following example:

```
String cmd = "UPDATE \"CData\".\"Sample\".Products SET Name='John' WHERE
Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete information from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause, as shown in the following example:

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]
<expression> ::=
    | @ <parameter>
    | ?
    | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows, as shown in the following example:

```
Connection connection = DriverManager.getConnection
("jdbc:cassandra:Database=MyCassandraDB;Port=9042;Server=127.0.0.1;");
String cmd = "DELETE FROM \"CData\".\"Sample\".Products WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements.

EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
    [ @ ] <input_name> = <expression>
} [ , ... ]
<expression> ::=
    | @ <parameter>
    | ?
    | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

PIVOT and UNPIVOT

PIVOT and **UNPIVOT** can be used to change a table-valued expression into another table.

PIVOT

PIVOT rotates a table-value expression by turning unique values from one column into multiple columns in the output. PIVOT can run aggregations where required on any column value.

PIVOT Syntax

```
"SELECT 'AverageCost' AS Cost_Sorted_By_Production_Days, [0], [1], [2],  
[3], [4]  
FROM  
(  
  SELECT DaysToManufacture, StandardCost  
  FROM Production.Product  
) AS SourceTable  
PIVOT  
(  
  AVG(StandardCost)  
  FOR DaysToManufacture IN ([0], [1], [2], [3], [4])  
) AS PivotTable;"
```

UNPIVOT

UNPIVOT carries out nearly the opposite to PIVOT by rotating columns of a table-valued expressions into column values.

UNPIVOT Syntax

```
"SELECT VendorID, Employee, Orders
FROM
(SELECT VendorID, Emp1, Emp2, Emp3, Emp4, Emp5
FROM pvt) p
UNPIVOT
(Orders FOR Employee IN
(Emp1, Emp2, Emp3, Emp4, Emp5)
)AS unpvt;"
```

For further information on PIVOT and UNPIVOT, see [FROM clause plus JOIN, APPLY, PIVOT \(Transact-SQL\)](#)

Connection String Options

The connection string properties are the various options that can be used to establish a connection. This section provides a complete list of the options you can configure in the connection string for this provider. Click the links for further details.

For more information on establishing a connection, see [Basic Tab](#).

Authentication

Property	Description
AuthScheme	The scheme used for authentication. Accepted entries are Basic, DSE, Kerberos, and LDAP.
Server	The host name or IP address of the server hosting the Cassandra database.
Port	The port for the Cassandra database.
LDAPServer	The host name or IP address of the LDAP server.
User	The Cassandra user account used to authenticate.
Password	The password used to authenticate the user.

LDAPPort	The port for the LDAP server.
Database	The name of the Cassandra keyspace.
DefaultLDAPUser	The default LDAP user used to connect to and communicate with the server, it must be set if the LDAP server do not allow anonymous bind.
LDAPPassword	The password of the default LDAP user. It must be set if the LDAP server do not allow anonymous bind.
SearchBase	The search base for your LDAPServer, used to look up users.
SearchFilter	The search filter for looking up usernames in LDAP. The default setting is (uid=), When using Active Directory set the filter to (sAMAccountName=).
UseSSL	This field sets whether SSL is enabled.

Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

SSH

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A private key to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert key if it has one.
SSHClientCertSubject	The subject of the SSH client certificate.
SSHClientCertType	The type of SSHClientCert private key.
SSHServer	The SSH server.
SSHPort	The SSH port.
SSHUser	The SSH user.

SSHPassword	The SSH password.
SSHServerFingerprint	The SSH server fingerprint.
UseSSH	Whether to tunnel the Cassandra connection over SSH. Use SSH.

Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

Logging

Property	Description
LogModules	Core modules to be included in the log file.

Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.

Miscellaneous

Property	Description
AggregationsSupported	Whether or not to support aggregations in the Cassandra server. Note that in queries to the provider, you must use single quotes to define strings.
AllowFiltering	When true, slow-performing queries are processed on the server.
CaseSensitivity	Enable case sensitivity to the CQL sending to the server, if set to True, the identifiers in the CQL will be enclosed in double quotation marks.
ConsistencyLevel	The consistency level determines how many of the replicas of the data you are interacting with need to respond for the query to be considered a success.
FlattenArrays	By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.
FlattenObjects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
NullToUnset	Use unset instead of NULL in CQL query when performing INSERT operations.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Cassandra.

Readonly	You can use this property to enforce read-only access to Cassandra from the provider.
RowScanDepth	The maximum number of rows to scan to look for the columns available in a table.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseJsonFormat	Whether to submit and return the JSON encoding for CQL data types.
UserDefinedViews	A filepath pointing to the JSON configuration file containing your custom views.
VarintToString	Map Cassandra VARINT to String value.

Authentication

This section provides a complete list of the Authentication properties you can configure in the connection string for this provider.

Property	Description
AuthScheme	The scheme used for authentication. Accepted entries are Basic, DSE, Kerberos, and LDAP.
Server	The host name or IP address of the server hosting the Cassandra database.
Port	The port for the Cassandra database.
LDAPServer	The host name or IP address of the LDAP server.
User	The Cassandra user account used to authenticate.
Password	The password used to authenticate the user.

LDAPPort	The port for the LDAP server.
Database	The name of the Cassandra keyspace.
DefaultLDAPUser	The default LDAP user used to connect to and communicate with the server, it must be set if the LDAP server do not allow anonymous bind.
LDAPPassword	The password of the default LDAP user. It must be set if the LDAP server do not allow anonymous bind.
SearchBase	The search base for your LDAPServer, used to look up users.
SearchFilter	The search filter for looking up usernames in LDAP. The default setting is (uid=), When using Active Directory set the filter to (sAMAccountName=).
UseSSL	This field sets whether SSL is enabled.

AuthScheme

The scheme used for authentication. Accepted entries are Basic, DSE, Kerberos, and LDAP.

Possible Values

Basic, DSE, Kerberos, LDAP

Data Type

string

Default Value

"Basic"

Remarks

Set this property to authenticate to open-source or DataStax Enterprise (DSE) Cassandra instances.

Together with [Password](#) and [User](#), this field is used to authenticate against the server. Basic is the default option. Use the following options to select your authentication scheme:

- Basic: Set this to authenticate with login credentials and Cassandra's built-in authentication.
- DSE: Set this to authenticate with login credentials and the DSE Unified Authenticator.
- Kerberos: Set this to use Kerberos to authenticate.
- LDAP: Set this to use LDAP to authenticate.

See the Getting Started section for guides to using each authentication method.

Server

The host name or IP address of the server hosting the Cassandra database.

Data Type

string

Default Value

""

Remarks

The host name or IP address of the server hosting the Cassandra database. To connect to a distributed system, you can set [Server](#) to a comma-separated list of servers and ports, separated by colons. You will also need to set [ConsistencyLevel](#).

Note that you must specify all of the servers required by your selected consistency level.

Port

The port for the Cassandra database.

Data Type

string

Default Value

"9042"

Remarks

The port for the Cassandra database.

LDAPServer

The host name or IP address of the LDAP server.

Data Type

string

Default Value

""

Remarks

The host name or IP address of the LDAP server.

User

The Cassandra user account used to authenticate.

Data Type

string

Default Value

""

Remarks

Together with [Password](#), this field is used to authenticate against the Cassandra server.

Password

The password used to authenticate the user.

Data Type

string

Default Value

""

Remarks

The [User](#) and [Password](#) are together used to authenticate with the server.

LDAPPort

The port for the LDAP server.

Data Type

string

Default Value

"389"

Remarks

The port for the LDAP server.

Database

The name of the Cassandra keyspace.

Data Type

string

Default Value

""

Remarks

The name of the Cassandra keyspace containing the tables.

DefaultLDAPUser

The default LDAP user used to connect to and communicate with the server, it must be set if the LDAP server do not allow anonymous bind.

Data Type

string

Default Value

""

Remarks

Specify the default LDAP user in case the LDAP server do not allow anonymous login.

LDAPPassword

The password of the default LDAP user. It must be set if the LDAP server do not allow anonymous bind.

Data Type

string

Default Value

""

Remarks

Specify the password of the default LDAP user.

SearchBase

The search base for your LDAPServer, used to look up users.

Data Type

string

Default Value

""

Remarks

The search base for your LDAPServer, used to look up users.

SearchFilter

The search filter for looking up usernames in LDAP. The default setting is (uid=), When using Active Directory set the filter to (sAMAccountName=).

Data Type

string

Default Value

"uid="

Remarks

The search filter for looking up usernames in LDAP. The default setting is (uid=).

UseSSL

This field sets whether SSL is enabled.

Data Type

bool

Default Value

false

Remarks

This field sets whether the adapter will attempt to negotiate TLS/SSL connections to the server. By default, the adapter checks the server's certificate against the system's trusted certificate store. To specify another certificate, set [SSLServerCert](#).

Kerberos

This section provides a complete list of the Kerberos properties you can configure in the connection string for this provider.

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

KerberosKDC

The Kerberos Key Distribution Center (KDC) service used to authenticate the user.

Data Type

string

Default Value

""

Remarks

The Kerberos properties are used when using SPNEGO or Windows Authentication. The adapter will request session tickets and temporary session keys from the Kerberos KDC service. The Kerberos KDC service is conventionally colocated with the domain controller.

If Kerberos KDC is not specified, the adapter will attempt to detect these properties automatically from the following locations:

- **KRB5 Config File (krb5.ini/krb5.conf):** If the KRB5_CONFIG environment variable is set and the file exists, the adapter will obtain the KDC from the specified file. Otherwise, it will attempt to read from the default MIT location based on the OS: `C:\ProgramData\MIT\Kerberos5\krb5.ini` (Windows) or `/etc/krb5.conf` (Linux).
- **Java System Properties:** Using the system properties `java.security.krb5.realm` and `java.security.krb5.kdc`.
- **Domain Name and Host:** If the Kerberos Realm and Kerberos KDC could not be inferred from another location, the adapter will infer them from the configured domain name and host.

Note: Windows authentication is supported in JRE 1.6 and above only.

KerberosRealm

The Kerberos Realm used to authenticate the user.

Data Type

string

Default Value

""

Remarks

The Kerberos properties are used when using SPNEGO or Windows Authentication. The Kerberos Realm is used to authenticate the user with the Kerberos Key Distribution Service (KDC). The Kerberos Realm can be configured by an administrator to be any string, but conventionally it is based on the domain name.

If Kerberos Realm is not specified, the adapter will attempt to detect these properties automatically from the following locations:

- **KRB5 Config File (krb5.ini/krb5.conf):** If the KRB5_CONFIG environment variable is set and the file exists, the adapter will obtain the default realm from the specified file. Otherwise, it will attempt to read from the default MIT location based on the OS: `C:\ProgramData\MIT\Kerberos5\krb5.ini` (Windows) or `/etc/krb5.conf` (Linux)
- **Java System Properties:** Using the system properties `java.security.krb5.realm` and `java.security.krb5.kdc`.

- **Domain Name and Host:** If the Kerberos Realm and Kerberos KDC could not be inferred from another location, the adapter will infer them from the user-configured domain name and host. This might work in some Windows environments.

Note: Kerberos-based authentication is supported in JRE 1.6 and above only.

KerberosSPN

The service principal name (SPN) for the Kerberos Domain Controller.

Data Type

string

Default Value

""

Remarks

If the SPN on the Kerberos Domain Controller is not the same as the URL that you are authenticating to, use this property to set the SPN.

KerberosKeytabFile

The Keytab file containing your pairs of Kerberos principals and encrypted keys.

Data Type

string

Default Value

""

Remarks

The Keytab file containing your pairs of Kerberos principals and encrypted keys.

KerberosServiceRealm

The Kerberos realm of the service.

Data Type

string

Default Value

""

Remarks

The KerberosServiceRealm is the specify the service Kerberos realm when using cross-realm Kerberos authentication.

In most cases, a single realm and KDC machine are used to perform the Kerberos authentication and this property is not required.

This property is available for complex setups where a different realm and KDC machine are used to obtain an authentication ticket (AS request) and a service ticket (TGS request).

KerberosServiceKDC

The Kerberos KDC of the service.

Data Type

string

Default Value

""

Remarks

The KerberosServiceKDC is used to specify the service Kerberos KDC when using cross-realm Kerberos authentication.

In most cases, a single realm and KDC machine are used to perform the Kerberos authentication and this property is not required.

This property is available for complex setups where a different realm and KDC machine are used to obtain an authentication ticket (AS request) and a service ticket (TGS request).

KerberosTicketCache

The full file path to an MIT Kerberos credential cache file.

Data Type

string

Default Value

""

Remarks

This property can be set if you wish to use a credential cache file that was created using the MIT Kerberos Ticket Manager or kinit command.

SSL

This section provides a complete list of the SSL properties you can configure in the connection string for this provider.

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.

SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

SSLClientCert

The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).

Data Type

string

Default Value

""

Remarks

The name of the certificate store for the client certificate.

The [SSLClientCertType](#) field specifies the type of the certificate store specified by [SSLClientCert](#). If the store is password protected, specify the password in [SSLClientCertPassword](#).

[SSLClientCert](#) is used in conjunction with the [SSLClientCertSubject](#) field in order to specify client certificates. If [SSLClientCert](#) has a value, and [SSLClientCertSubject](#) is set, a search for a certificate is initiated. See [SSLClientCertSubject](#) for more information.

Designations of certificate stores are platform-dependent.

The following are designations of the most common User and Machine certificate stores in Windows:

MY	A certificate store holding personal certificates with their associated private keys.
CA	Certifying authority certificates.

ROOT	Root certificates.
SPC	Software publisher certificates.

In Java, the certificate store normally is a file containing certificates and optional private keys.

When the certificate store type is PFXFile, this property must be set to the name of the file. When the type is PFXBlob, the property must be set to the binary contents of a PFX file (for example, PKCS12 certificate store).

SSLClientCertType

The type of key store containing the TLS/SSL client certificate.

Possible Values

USER, MACHINE, PFXFILE, PFXBLOB, JKSFIL, JKSBLOB, PEMKEY_FILE, PEMKEY_BLOB, PUBLIC_KEY_FILE, PUBLIC_KEY_BLOB, SSHPUBLIC_KEY_FILE, SSHPUBLIC_KEY_BLOB, P7BFILE, PPKFILE, XMLFILE, XMLBLOB

Data Type

string

Default Value

"USER"

Remarks

This property can take one of the following values:

USER - default	For Windows, this specifies that the certificate store is a certificate store owned by the current user. Note that this store type is not available in Java.
MACHINE	For Windows, this specifies that the certificate store is a machine

	store. Note that this store type is not available in Java.
PFXFILE	The certificate store is the name of a PFX (PKCS12) file containing certificates.
PFXBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in PFX (PKCS12) format.
JKSFILE	The certificate store is the name of a Java key store (JKS) file containing certificates. Note that this store type is only available in Java.
JKSBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in JKS format. Note that this store type is only available in Java.
PEMKEY_FILE	The certificate store is the name of a PEM-encoded file that contains a private key and an optional certificate.
PEMKEY_BLOB	The certificate store is a string (base64-encoded) that contains a private key and an optional certificate.
PUBLIC_KEY_FILE	The certificate store is the name of a file that contains a PEM- or DER-encoded public key certificate.
PUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains a PEM- or DER-encoded public key certificate.
SSHPUBLIC_KEY_FILE	The certificate store is the name of a file that contains an SSH-style public key.
SSHPUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains an SSH-style public key.
P7BFILE	The certificate store is the name of a PKCS7 file containing certificates.
PPKFILE	The certificate store is the name of a file that contains a PuTTY Private Key (PPK).

XMLFILE	The certificate store is the name of a file that contains a certificate in XML format.
XMLBLOB	The certificate store is a string that contains a certificate in XML format.

SSLClientCertPassword

The password for the TLS/SSL client certificate.

Data Type

string

Default Value

""

Remarks

If the certificate store is of a type that requires a password, this property is used to specify that password to open the certificate store.

SSLClientCertSubject

The subject of the TLS/SSL client certificate.

Data Type

string

Default Value

"*"

Remarks

When loading a certificate the subject is used to locate the certificate in the store.

If an exact match is not found, the store is searched for subjects containing the value of the property. If a match is still not found, the property is set to an empty string, and no certificate is selected.

The special value "*" picks the first certificate in the certificate store.

The certificate subject is a comma separated list of distinguished name fields and values. For example, "CN=www.server.com, OU=test, C=US, E=support@company.com". The common fields and their meanings are shown below.

Field	Meaning
CN	Common Name. This is commonly a host name like www.server.com.
O	Organization
OU	Organizational Unit
L	Locality
S	State
C	Country
E	Email Address

If a field value contains a comma, it must be quoted.

SSLServerCert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine is rejected.

This property can take the following forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIICHTCCAe4CAQAwDQYJKoZIhvd.....Qw == -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	34e929226ae0819f2ec14b4a3d904f801c
The SHA1 Thumbprint (hex values can also be either space or colon separated)	bb150d

If not specified, any certificate trusted by the machine is accepted.

Certificates are validated as trusted by the machine based on the System's trust store. The trust store used is the 'javax.net.ssl.trustStore' value specified for the system. If no value is specified for this property, Java's default trust store is used (for example, JAVA_HOME\lib\security\cacerts).

Use '*' to signify to accept all certificates. Note that this is not recommended due to security concerns.

SSH

This section provides a complete list of the SSH properties you can configure in the connection string for this provider.

Property	Description
SSHAuthMode	The authentication method to be used to log on to an SFTP server.
SSHClientCert	A private key to be used for authenticating the user.
SSHClientCertPassword	The password of the SSHClientCert key if it has one.
SSHClientCertSubject	The subject of the SSH client certificate.
SSHClientCertType	The type of SSHClientCert private key.
SSHServer	The SSH server.
SSHPort	The SSH port.
SSHUser	The SSH user.
SSHPassword	The SSH password.
SSHServerFingerprint	The SSH server fingerprint.
UseSSH	Whether to tunnel the Cassandra connection over SSH. Use SSH.

SSHAuthMode

The authentication method to be used to log on to an SFTP server.

Possible Values

None, Password, Public_Key

Data Type

string

Default Value

"Password"

Remarks

- None: No authentication will be performed. The current [User](#) value is ignored, and the connection will be logged in as anonymous.
- Password: The adapter will use the values of [User](#) and [Password](#) to authenticate the user.
- Public_Key: The adapter will use the values of [User](#) and [SSHClientCert](#) to authenticate the user. [SSHClientCert](#) must have a private key available for this authentication method to succeed.

SSHClientCert

A private key to be used for authenticating the user.

Data Type

string

Default Value

""

Remarks

[SSHClientCert](#) must contain a valid private key in order to use public key authentication. A public key is optional, if one is not included then the adapter generates it from the private key. The adapter sends the public key to the server and the connection is allowed if the user has authorized the public key.

The [SSHClientCertType](#) field specifies the type of the key store specified by [SSHClientCert](#). If the store is password protected, specify the password in [SSHClientCertPassword](#).

Some types of key stores are containers which may include multiple keys. By default the adapter will select the first key in the store, but you can specify a specific key using [SSHClientCertSubject](#).

SSHClientCertPassword

The password of the SSHClientCert key if it has one.

Data Type

string

Default Value

""

Remarks

This property is only used when authenticating to SFTP servers with [SSHAuthMode](#) set to PublicKey and [SSHClientCert](#) set to a private key.

SSHClientCertSubject

The subject of the SSH client certificate.

Data Type

string

Default Value

"*"

Remarks

When loading a certificate the subject is used to locate the certificate in the store.

If an exact match is not found, the store is searched for subjects containing the value of the property.

If a match is still not found, the property is set to an empty string, and no certificate is selected.

The special value "*" picks the first certificate in the certificate store.

Remarks

This property can take one of the following values:

Types	Description	Allowed Blob Values
MACHINE/USER	Not available on this platform.	Blob values are not supported.
JKSFILE/JKSBLOB	A Java keystore file. Must contain both a certificate and a private key. Only available in Java.	base64-only
PFXFILE/PFXBLOB	A PKCS12-format (.pfx) file. Must contain both a certificate and a private key.	base64-only
PEMKEY_FILE/PEMKEY_BLOB	A PEM-format file. Must contain an RSA, DSA, or OPENSSH private key. Can optionally contain a certificate matching the private key.	base64 or plain text. Newlines may be replaced with spaces when providing the blob as text.
PPKFILE/PPKBLOB	A PuTTY-format private key created using the <i>puttygen</i> tool.	base64-only
XMLFILE/XMLBLOB	An XML key in the format generated by the .NET RSA class: <i>RSA.ToXmlString(true)</i> .	base64 or plain text.

SSHServer

The SSH server.

Data Type

string

Default Value

""

Remarks

The SSH server.

SSHPort

The SSH port.

Data Type

string

Default Value

"22"

Remarks

The SSH port.

SSHUser

The SSH user.

Data Type

string

Default Value

""

Remarks

The SSH user.

SSHPassword

The SSH password.

Data Type

string

Default Value

""

Remarks

The SSH password.

SSHServerFingerprint

The SSH server fingerprint.

Data Type

string

Default Value

""

Remarks

The SSH server fingerprint.

UseSSH

Whether to tunnel the Cassandra connection over SSH. Use SSH.

Data Type

bool

Default Value

false

Remarks

By default the adapter will attempt to connect directly to Cassandra. When this option is enabled, the adapter will instead establish an SSH connection with the [SSHServer](#) and tunnel the connection to Cassandra through it.

Firewall

This section provides a complete list of the Firewall properties you can configure in the connection string for this provider.

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

FirewallType

The protocol used by a proxy-based firewall.

Possible Values

NONE, TUNNEL, SOCKS4, SOCKS5

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the [FirewallServer](#) proxy.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Cassandra and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort . If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

FirewallServer

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by [FirewallType](#): Use [FirewallServer](#) with this property to connect through SOCKS or do tunneling.

FirewallPort

The TCP port for a proxy-based firewall.

Data Type

int

Default Value

0

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use [FirewallServer](#) to specify the name or IP address. Specify the protocol with [FirewallType](#).

FirewallUser

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The [FirewallUser](#) and [FirewallPassword](#) properties are used to authenticate against the proxy specified in [FirewallServer](#) and [FirewallPort](#), following the authentication method specified in [FirewallType](#).

FirewallPassword

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by [FirewallServer](#) and [FirewallPort](#), following the authentication method specified by [FirewallType](#).

Logging

This section provides a complete list of the Logging properties you can configure in the connection string for this provider.

Property	Description
LogModules	Core modules to be included in the log file.

LogModules

Core modules to be included in the log file.

Data Type

string

Default Value

""

Remarks

Only the modules specified (separated by ';') will be included in the log file. By default all modules are included.

See the [Logging](#) page for an overview.

Schema

This section provides a complete list of the Schema properties you can configure in the connection string for this provider.

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

"%APPDATA%\\CData\\Cassandra Data Provider\\Schema"

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The folder location can be a relative path from the location of the executable. The Location property is only needed if you want to customize definitions (for example, change a column name, ignore a column, and so on) or extend the data model with new tables, views, or stored procedures.

If left unspecified, the default location is "%APPDATA%\\CData\\Cassandra Data Provider\\Schema" with %**APPDATA**% being set to the user's configuration directory:

Platform	%APPDATA%
Windows	The value of the APPDATA environment variable
Mac	~/Library/Application Support
Linux	~/.config

Miscellaneous

This section provides a complete list of the Miscellaneous properties you can configure in the connection string for this provider.

Property	Description
AggregationsSupported	Whether or not to support aggregations in the Cassandra server. Note that in queries to the provider, you must use single quotes

	to define strings.
AllowFiltering	When true, slow-performing queries are processed on the server.
CaseSensitivity	Enable case sensitivity to the CQL sending to the server, if set to True, the identifiers in the CQL will be enclosed in double quotation marks.
ConsistencyLevel	The consistency level determines how many of the replicas of the data you are interacting with need to respond for the query to be considered a success.
FlattenArrays	By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.
FlattenObjects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
NullToUnset	Use unset instead of NULL in CQL query when performing INSERT operations.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Cassandra.
Readonly	You can use this property to enforce read-only access to Cassandra from the provider.
RowScanDepth	The maximum number of rows to scan to look for the columns available in a table.
Timeout	The value in seconds until the timeout error is thrown, canceling

	the operation.
UseJsonFormat	Whether to submit and return the JSON encoding for CQL data types.
UserDefinedViews	A filepath pointing to the JSON configuration file containing your custom views.
VarintToString	Map Cassandra VARINT to String value.

AggregationsSupported

Whether or not to support aggregations in the Cassandra server. Note that in queries to the provider, you must use single quotes to define strings.

Data Type

bool

Default Value

true

Remarks

AllowFiltering

When true, slow-performing queries are processed on the server.

Data Type

bool

Default Value

false

Remarks

Cassandra by default does not allow filtering for queries that it predicts will have performance problems. These queries include filtering on a column that is not the primary key.

You can override the default behavior and rely on the server to process these queries by setting `AllowFiltering` to true.

CaseSensitivity

Enable case sensitivity to the CQL sending to the server, if set to True, the identifiers in the CQL will be enclosed in double quotation marks.

Data Type

bool

Default Value

true

Remarks

By default, SQL is case-insensitive. However, Cassandra supports case-sensitive table and column names. Setting this property to True will enable you to retrieve tables and columns based on their case-sensitive names.

ConsistencyLevel

The consistency level determines how many of the replicas of the data you are interacting with need to respond for the query to be considered a success.

Possible Values

ONE, TWO, THREE, QUORUM, ALL, LOCAL_QUORUM, EACH_QUORUM, SERIAL, LOCAL_SERIAL, LOCAL_ONE, ANY

Data Type

string

Default Value

"ONE"

Remarks

The consistency level determines how many of the replicas of the data you are interacting with need to respond for the query to be considered a success. You need to specify the appropriate replicas in the [Server](#) property.

Below are the possible values:

- ANY: At least one replica must return success in a write operation. This property guarantees that a write never fails; this consistency level delivers the lowest consistency and highest availability.
- ALL: All replicas must respond. This property provides the highest consistency and the lowest availability.
- ONE: At least one replica must respond. This is the default and suitable for most users, who do not typically require high consistency.
- TWO: At least two replicas must respond.
- THREE: At least three replicas must respond.
- QUORUM: A quorum of nodes must respond. The QUORUM properties provide high consistency with some failure tolerance.
- EACH_QUORUM: A quorum of nodes must respond where a quorum is calculated for each data center. This setting maintains consistency in each data center.
- SERIAL: A quorum of replicas performs a consensus algorithm to allow lightweight transactions.
- LOCAL_ONE: At least one replica in the local data center must respond.
- LOCAL_SERIAL: The consensus algorithm is calculated for the local data center.
- LOCAL_QUORUM: A quorum of nodes must respond where the quorum is calculated for the local data center.

FlattenArrays

By default, nested arrays are returned as strings of JSON. The `FlattenArrays` property can be used to flatten the elements of nested arrays into columns of their own. Set `FlattenArrays` to the number of elements you want to return from nested arrays.

Data Type

string

Default Value

""

Remarks

By default, nested arrays are returned as strings of JSON. The `FlattenArrays` property can be used to flatten the elements of nested arrays into columns of their own. This is only recommended for arrays that are expected to be short.

Set `FlattenArrays` to the number of elements you want to return from nested arrays. The specified elements are returned as columns. The zero-based index is concatenated to the column name. Other elements are ignored.

For example, you can return an arbitrary number of elements from an array of strings:

```
[ "FLOW-MATIC", "LISP", "COBOL" ]
```

When `FlattenArrays` is set to 1, the preceding array is flattened into the following table:

Column Name	Column Value
languages_0	FLOW-MATIC

FlattenObjects

Set `FlattenObjects` to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.

Data Type

bool

Default Value

false

Remarks

Set `FlattenObjects` to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON. The property name is concatenated onto the object name with an underscore to generate the column name.

For example, you can flatten the nested objects below at connection time:

```
[
  { "grade": "A", "score": 2 },
  { "grade": "A", "score": 6 },
  { "grade": "A", "score": 10 },
  { "grade": "A", "score": 9 },
  { "grade": "B", "score": 14 }
]
```

When `FlattenObjects` is set to true and `FlattenArrays` is set to 1, the preceding array is flattened into the following table:

Column Name	Column Value
grades_0_grade	A
grades_0_score	2

MaxRows

Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.

Data Type

int

Default Value

-1

Remarks

Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.

NullToUnset

Use unset instead of NULL in CQL query when performing INSERT operations.

Data Type

bool

Default Value

false

Remarks

In Cassandra 2.2 and above, when executing an INSERT query, a parameter value can be set to unset. Cassandra does not consider unset field values which helps to avoid tombstones.

When NULL values are inserted, it is possible to reach the tombstone threshold limits which causes an exception to be thrown when querying the data. Setting this property to true and submitting unset values avoids these tombstones from being created.

Note: This option is only available on INSERT operations as Cassandra does not support changing existing values to unset.

Other

These hidden properties are used only in specific use cases.

Data Type

string

Default Value

""

Remarks

The properties listed below are available for specific use cases. Normal driver use cases and functionality should not require these properties.

Specify multiple properties in a semicolon-separated list.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Determines whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Pagesize

The maximum number of results to return per page from Cassandra.

Data Type

int

Default Value

5000

Remarks

The Pagesize property affects the maximum number of results to return per page from Cassandra. Setting a higher value may result in better performance at the cost of additional memory allocated per page consumed.

Readonly

You can use this property to enforce read-only access to Cassandra from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

RowScanDepth

The maximum number of rows to scan to look for the columns available in a table.

Data Type

int

Default Value

100

Remarks

The columns in a table must be determined by scanning table rows. This value determines the maximum number of rows that will be scanned.

Setting a high value may decrease performance. Setting a low value may prevent the data type from being determined properly, especially when there is null data.

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

int

Default Value

60

Remarks

If Timeout = 0, operations do not time out. The operations run until they complete successfully or until they encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

UseJsonFormat

Whether to submit and return the JSON encoding for CQL data types.

Data Type

bool

Default Value

true

Remarks

Cassandra 2.2 introduced a CQL extension that allows you to JSON-encode CQL data types. By default, you use the JSON syntax to manipulate data and SELECT statements return JSON through the adapter. Set this property to false to use CQL literals to interact with Cassandra data.

The syntax for CQL literals has several differences from JSON. For example:

- CQL strings are defined in single quotes, while JSON strings are defined in double quotes.
- CQL sets, tuples, and lists are JSON-encoded as arrays.
- User-defined types and CQL *uuid* types are JSON-encoded as objects.

Refer to the CQL documentation for more information on how to JSON-encode data types in your version of Cassandra. Below is an example SQL statement using JSON and CQL.

Format	Syntax
CQL	<pre>INSERT INTO users (user_id, emails) VALUES(@user_id, @emails)</pre>
Parameters	
user_id	frodo
emails	{'f@baggins.com', 'baggins@gmail.com'}
JSON	<pre>INSERT INTO users (user_id, emails) VALUES (@user_id, @emails)</pre>
Parameters	
user_id	frodo

emails	["f@baggins.com", "baggins@gmail.com"])
--------	--

Note that in queries to the adapter, you must use single quotes to define strings.

UserDefinedViews

A filepath pointing to the JSON configuration file containing your custom views.

Data Type

string

Default Value

""

Remarks

User Defined Views are defined in a JSON-formatted configuration file called *UserDefinedViews.json*. The adapter automatically detects the views specified in this file.

You can also have multiple view definitions and control them using the [UserDefinedViews](#) connection property. When you use this property, only the specified views are seen by the adapter.

This User Defined View configuration file is formatted as follows:

- Each root element defines the name of a view.
- Each root element contains a child element, called **query**, which contains the custom SQL query for the view.

For example:

```
{
  "MyView": {
    "query": "SELECT * FROM \"CData\\\".\"Sample\\\".Products WHERE MyColumn =
'value'"
  },
  "MyView2": {
```

```

    "query": "SELECT * FROM MyTable WHERE Id IN (1,2,3)"
  }
}

```

Use the `UserDefinedViews` connection property to specify the location of your JSON configuration file. For example:

```

"UserDefinedViews",
"C:\\Users\\yourusername\\Desktop\\tmp\\UserDefinedViews.json"

```

VarintToString

Map Cassandra VARINT to String value.

Data Type

bool

Default Value

true

Remarks

Map Cassandra VARINT to String value.

TIBCO Product Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join the TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the [TIBCO Product Documentation](#) website, mainly in HTML and PDF formats.

The [TIBCO Product Documentation](#) website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The following documentation for this product is available on the [TIBCO® Data Virtualization](#) page.

- **Users**
 - TDV Getting Started Guide
 - TDV User Guide
 - TDV Web UI User Guide
 - TDV Client Interfaces Guide
 - TDV Tutorial Guide
 - TDV Northbay Example
- **Administration**
 - TDV Installation and Upgrade Guide
 - TDV Administration Guide
 - TDV Active Cluster Guide
 - TDV Security Features Guide
- **Data Sources**

TDV Adapter Guides

TDV Data Source Toolkit Guide (Formerly Extensibility Guide)

- **References**

TDV Reference Guide

TDV Application Programming Interface Guide

- **Other**

TDV Business Directory Guide

TDV Discovery Guide

- *TIBCO TDV and Business Directory Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

How to Contact TIBCO Support

Get an overview of [TIBCO Support](#). You can contact TIBCO Support in the following ways:

- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the [TIBCO Support](#) website.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to [TIBCO Support](#) website. If you do not have a user name, you can request one by clicking **Register** on the website.

Release Version Support

TDV 8.5 is designated as a Long Term Support (LTS) version. Some release versions of TIBCO® Data Virtualization products are selected to be long-term support (LTS) versions. Defect corrections will typically be delivered in a new release version and as hotfixes or service packs to one or more LTS versions. See also

https://docs.tibco.com/pub/tdv/general/LTS/tdv_LTS_releases.htm.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](#). For a free registration, visit [TIBCO Community](#).

Legal and Third-Party Notices

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, TIBCO logo, TIBCO O logo, ActiveSpaces, Enterprise Messaging Service, Spotfire, TERR, S-PLUS, and S+ are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle Corporation and/or its affiliates.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the

readme file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

This and other products of TIBCO Software Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2002-2023 Cloud Software Group, Inc All Rights Reserved.