



DataStax DevCenter

Documentation

January 21, 2016

Contents

Features..... 3

Installing DevCenter..... 5

Using SSL connections.....5

Reference.....7

 GUI.....7

 Connection Manager..... 8

 Query Editor.....14

 Schema Navigator..... 19

 Outline.....55

 Results.....55

 Query Trace.....56

 CQL Scripts..... 58

 Automatic updates.....60

 Submit Feedback form..... 61

 Working with JSON data.....62

 Keyboard shortcuts.....66

 Usage data..... 67

 User-defined functions and aggregate functions..... 70

FAQ.....71

Using the docs.....72

Features

DataStax DevCenter is a free visual schema and query IDE for developers, administrators, and others who want to create and run Cassandra Query Language (CQL) statements against Apache Cassandra and DataStax Enterprise. Users can quickly add and create new connections, import previously saved queries, and navigate database instances. Tabbed editors allow administrators to work on multiple database sessions at a time and compare results.

What's new in 1.5?

- Support for Cassandra 3.0 features:
 - New, Edit, Clone, and Drop [Materialized View wizards](#)
 - Multiple secondary indexes
 - Syntax highlighting, code snippets and code assist support for materialized views in the CQL editor
- **Schema Navigator** improvements:
 - **Materialized View** nodes are displayed both under their corresponding keyspace and also under their base table
 - Maintains the state of expanded nodes and also the position on a refresh
 - Displays only those schema elements supported by the version of Cassandra in the cluster
- Drop schema objects:
 - Added wizards for dropping User-defined Functions and User-defined Aggregates
 - The drop action checks for dependent objects (for example, dropping a table checks for materialized views, dropping a User-defined type checks for table and User-defined types, dropping a User-defined function checks for User-defined aggregates)
- Improved timestamp formatting used in the **Query Results** and **Detailed Results** views; the new format is `yyyy-MM-dd hh:mm:ssZ`
- Editor and Wizard support in Cassandra 3.0 for new compression options `class` and `chunk_length_in_kb`, and for `crc_check_chance` as a table-level property
- Ability to [enable or disable the Query Trace](#) feature
- Improved content assist proposals for caching, compaction, compressions and clustering properties for `ALTER TABLE` statements
- Authentication against [DataStax Enterprise clusters configured to use LDAP](#) has been tested with the following LDAP providers: OpenLDAP, OracleLDAP, WinAD08, or WinAD12
- Help links in wizards use the underlying connection details to link to the version-specific documentation pages

Features in DevCenter 1.4.1

- Improved Index name validation for Cassandra 1.2 and 2.0
- Improved content assist and validation for `DROP FUNCTION` statement
- [New usage data metrics](#): DevCenter version, Java version, and OS version

Features in DevCenter 1.4

- New wizards
 - **New Index wizard**: allows you to create an index on a column
 - **New User-defined Type wizards**: allows you to create, edit, and clone your UDTs by using interactive wizards
- Support for Apache Cassandra 2.2 features:
 - Advanced [JSON](#) support in the editor and a new **Results** viewer

Features

- User-defined [functions](#) (UDF) and [aggregates](#) (UDA)
- New data types: date, time, smallint, and tinyint
- Role-based authentication CQL statements
- Secondary index support for map key-value pairs with `CREATE INDEX ... ENTRIES`
- Ability to drop database objects from the **Schema Navigator**
- An enhanced **Results** panel with a [Details data viewer](#), including TTL and write-time data
- Ability to [specify the consistency level](#) when executing queries
- [Automatic update](#) notifications when new DevCenter versions become available
- Collection and sending of [anonymous usage data](#)

Features in DevCenter 1.3.1

- Support for Cassandra 2.1.3 CQL features:
 - Frozen collections including nested collections and `FULL` collection indexes.
 - Inclusion of `IF EXISTS` for `UPDATE` statements.
- Editor and wizard support for the `DateTieredCompactionStrategy`.
- Additional table options available in the **Create** and **Edit Table** wizard:
 - `default_time_to_live`
 - `gc_grace_seconds`
 - `index_interval`
 - `min_index_interval`
 - `max_index_interval`
 - `memtable_flush_period_in_ms`
 - `populate_io_cache_on_flush`
 - `speculative_retry`

Features in DevCenter 1.3.0

- [Execute \(single or multiple\) selected statement\(s\)](#)
- Wizards
 - [Keyspace wizard](#)
 - [Table wizard](#)
- [A feedback form has been added](#)

Features in DevCenter 1.2.1

- **New features**
 - [Prevent execution of CQL scripts containing syntax errors](#)
 - *Copy as INSERT* now supports collections of UDTs or Tuples
- **Bug fixes**
 - Fix issue on Mac OS/X Yosemite which caused first row of results and query trace table to be hidden
 - Fix NPE appearing in log file on some content assist actions

Features in DevCenter 1.2

- Full support for Cassandra 2.1, including:
 - user-defined types (UDTs)
 - tuples
 - `IF EXISTS`
- Syntax highlighting, representation in **Results** tab and **Schema navigator**, validation, content assist, and code snippets for UDTs and tuples

- A new **Query Trace** tab, located next to the **Results** tab, which displays detailed trace event data for the last executed query to aid in understanding query execution and performance
- An improved **New Connection** wizard for creating and managing Cassandra connections

Features in DevCenter 1.1

- Support for Apache Cassandra 2.0.x and DataStax Enterprise 4.0.x (lightweight transactions syntax, static columns, `uuid()`, `now`, etc.)
- New and improved validation and code-assist rules
- Option to use a default keyspace for running a script
- Option to set the maximum number of rows to be returned by a statement
- Copy selected or all results as CSV or CQL inserts
- Option to enable SSL connection
- See [the posting on the DataStax Developer Blog](#) about the DevCenter 1.1 feature set

Features in DevCenter 1.0

- A smart CQL editor that provides
 - syntax highlighting
 - code auto-completion (both keyword and snippet)
 - real-time script validation against the current connection
- Schema explorer view for browsing
 - keyspaces
 - tables
 - other database objects
- Outline view for allowing quick navigating long CQL scripts
- Configuring connections to Cassandra or DataStax Enterprise clusters requiring authorization

Installing DevCenter

Prerequisites

DataStax DevCenter is a Java application and requires, at least JRE 6. The latest JRE is recommended.

Procedure

1. [Download](#) the version of DevCenter for your operating system.
2. Unzip the downloaded file to a location on your hard drive.
3. Run the executable file, `DevCenter`.

What to do next

[Take the DevCenter tutorial](#).

Using SSL connections

Prerequisites

- SSL must be configured and working on your cluster.
 - [Client-to-node encryption](#).
 - [Node-to-node encryption](#).
 - [Preparing server certificates](#).
- Install the Java Cryptography Extension (JSE) on your client system.

Download the same version of Java as Cassandra or DataStax Enterprise use:

- [Java 8](#)
- [Java 7](#)
- [Java 6](#)

Installation directory (jre lib/security):

- Linux: /usr/lib/jvm/jdk1.*major.minor_update*/jre/lib/security
- Mac OS X: /Library/Java/JavaVirtualMachines/jdk1.*major.minor_update*/Contents/Home/jre/lib/security
- Windows: C:\Program Files\Java\jre7\lib\security

Extract the downloaded file and copy the content of UnlimitedJCEPolicy directory to the jre/lib/security directory.

- The `keytool` command to manage encryption keys.

Note: If you cannot find the `keytool` command on a Windows system, read [these instructions](#).

Procedure

Server verification

1. To perform server verification, the client needs to have the public key certificate of each node in the cluster stored in a local truststore file. This file is password protected (`keytool` prompts to create a password). The truststore file and password is entered into the DevCenter connection manager dialog box (see below).

- a) Using `keytool`, create a truststore file on your client by importing the public key certificates from each node in your cluster.

```
$ keytool -import -v -trustcacerts -alias node0 -file node0.cer -  
keystore .truststore  
$ keytool -import -v -trustcacerts -alias node1 -file node1.cer -  
keystore .truststore  
$ keytool -import -v -trustcacerts -alias node2 -file node2.cer -  
keystore .truststore
```

- b) In DevCenter, select **File > New > Connection** to open the **Connection Manager**.
- c) Add the IP addresses of the nodes in your cluster.
- d) Select **Next**.
- e) Select **This cluster requires SSL** option and enter a full path to (or navigate to) the `truststore` file on your machine.
- f) Enter the truststore password.
- g) Select **Try to establish a connection** link to verify that you can successfully connect to Cassandra nodes.

Client verification

2. If your cluster requires client verification, you need to perform the following additional steps:
 - a) Create an SSL certificate for the client host (that is, the system on which DevCenter is installed).

```
$ keytool -genkey -alias client-host -keystore .keystore
```

- b) Export the client certificate.

```
$ keytool -export -alias client-host -file client-host.cer -
keystore .keystore
```

The public certificate is stored in the *client-host.cer* file.

- c) Copy the public certificate and import it into the truststore on all nodes of the Cassandra cluster which you want DevCenter to be connected to.

```
$ keytool -import -v -trustcacerts -alias client-host -file /tmp/client-
host.cer -keystore /var/tmp/.truststore
```

Note: You may have to ask your cluster administrator if you do not have the proper permissions to modify the truststore file on the cluster nodes.

- d) In DevCenter, right-click your connection and select **Properties** to edit the connection in the **Connection Manager**.
- e) In **Advanced Settings** (under **Basic Settings**) select the **Client authentication required** option and enter location of the keystore file and keystore password.
The **Connection manager** displays an error if the keystore filepath or password is incorrect.
- f) Click the **Try to establish a connection** link to verify your configuration.
- g) Click **OK** at the bottom of the **Connection Manager** dialog to create or update the connection.
Now you can enable and have DevCenter communicate with your SSL-enabled cluster.

Reference

GUI

The default main window for DevCenter has six panes arranged clockwise from the top left-hand corner:

Connection Manager

Creates, deletes, opens, and closes cluster connections.

Query Editor

A text editor for writing and executing CQL queries. When more than one file is open, it is a tabbed pane. The cluster against which the queries are executed appears in a dropdown.

Schema Navigator

Shows the schemas (keyspaces) of the currently active cluster.

Results

Shows the results of the last run query.

Query Trace

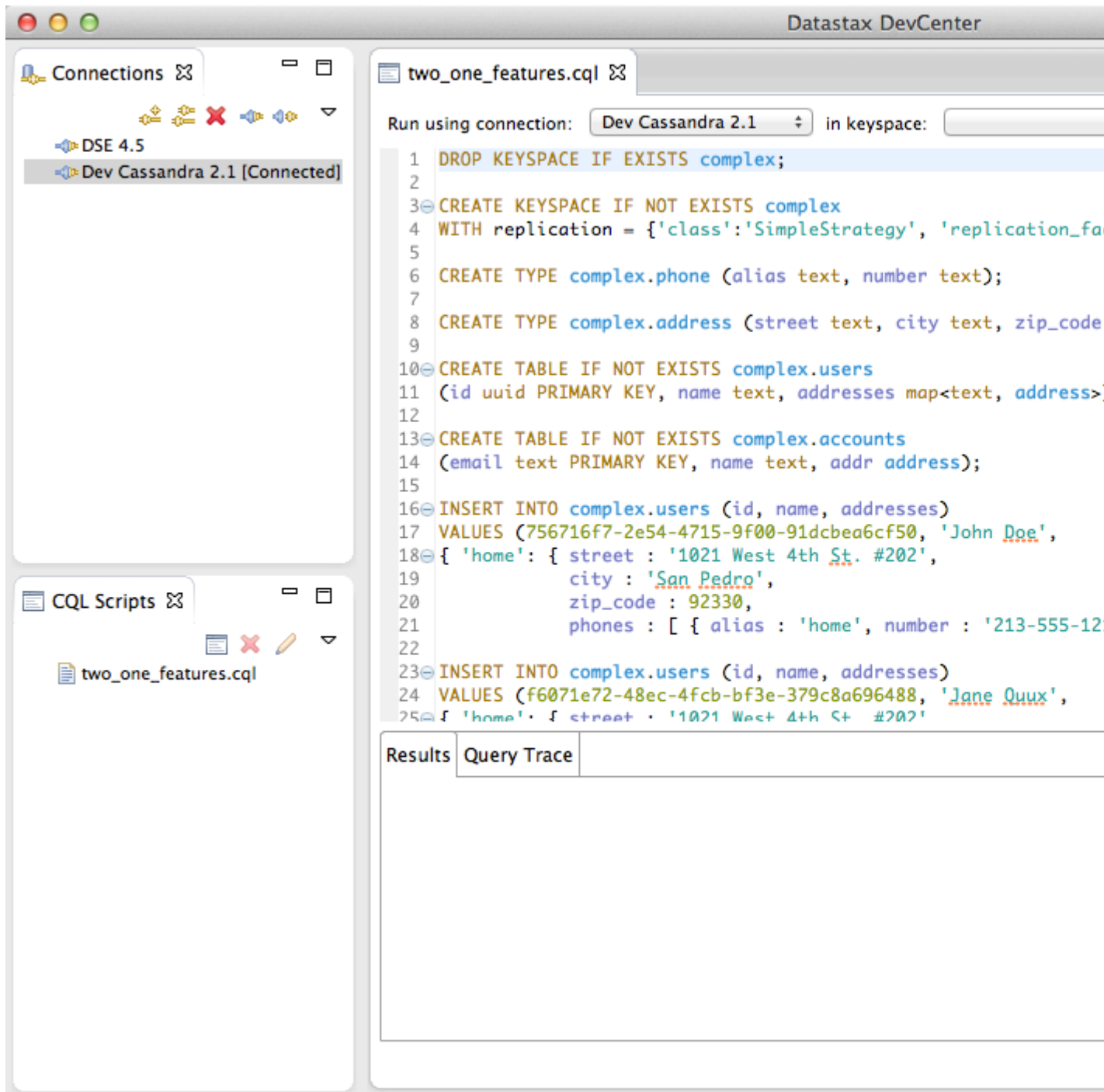
Shows the trace session of the last run query.

CQL Scripts

Creates, deletes, or opens for editing a CQL script.

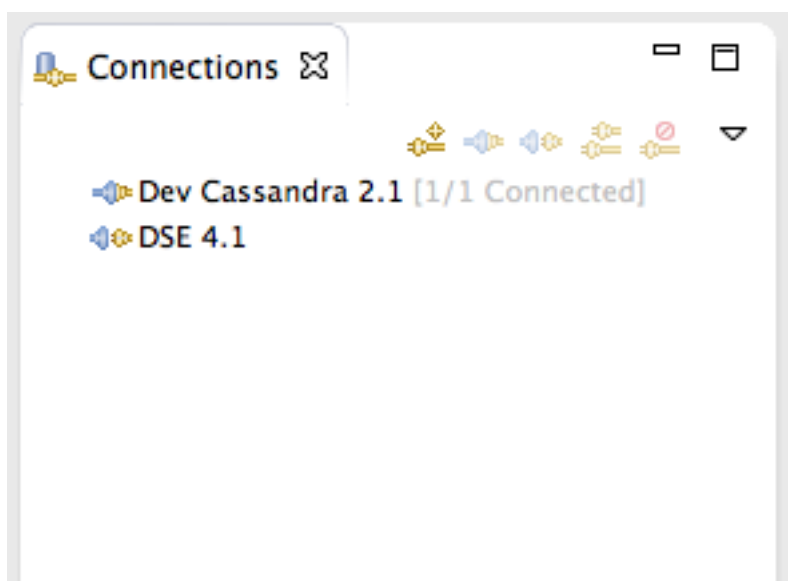
Outline

Shows the CQL statements in the currently open CQL script. Double-clicking takes you to the corresponding statement in the open file.



DevCenter is built using the Eclipse RCP (Rich Client Platform) and allows you to customize the panes organization as you prefer.

Connection Manager



When connected, the square brackets to the right of the connection name display **number-of-connected-nodes / number-of-nodes-in-hosts-list**.



New connection

Add a new connection by either selecting **File > New > Connection**, by clicking the **New connection** button in the toolbar of the connection manager view, or typing ###+N (Macintosh) / Ctrl+Alt+Shift+N (Windows, Linux).


The dialog for creating a new connection allows configuring a connection to an existing cluster by entering:

- a connection name
- one or more IP addresses of nodes in the cluster
- the connection protocol (by default 9042)
- which compression to use (none, Snappy, or LZ4)
- Security configuration:
 - whether the cluster is using authentication (either [internal](#) or [Kerberos](#)); if so, supply the authentication credentials
 - whether SSL is to be used for [node-to-node encryption](#)
 - The **Truststore file** contains the public certificates from the cluster nodes, and the Truststore password is used to access the password-protected Truststore file.
 - whether client authentication is to be used (that is, if `require_client_auth` is set to true in the `cassandra.yaml` file); supply the authentication credentials
 - The **Keystore file** contains the private certificate for the client, and the Keystore password is used to access the password-protected Keystore file.

New Connection

Basic Settings

Enter information about the cluster to connect to.



Connection name:

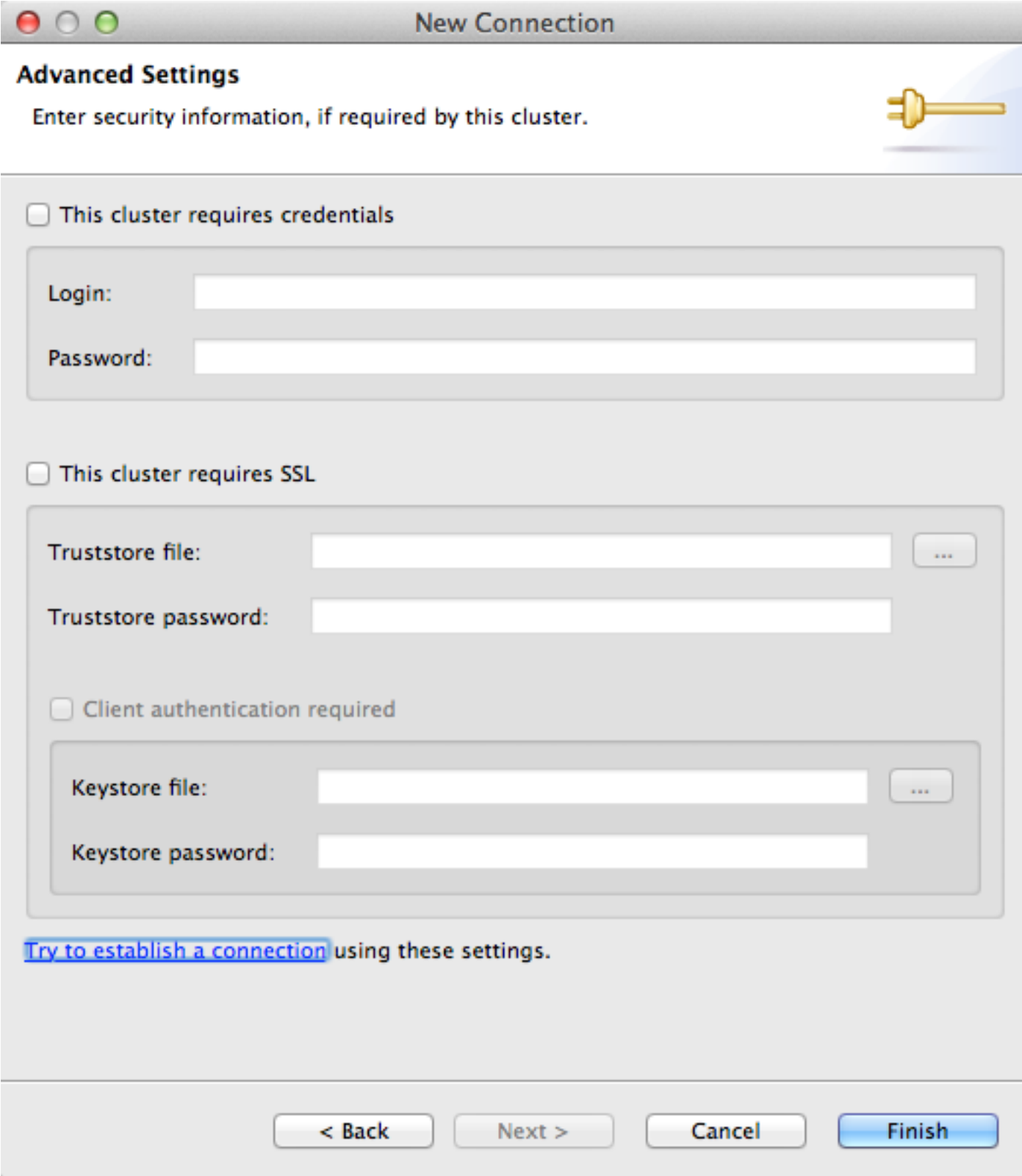
Contact hosts:

127.0.0.1

Native Protocol port:

Use compression: ☒ None ☐ Snappy ☐ LZ4

[Try to establish a connection](#) using these settings.



The image shows a 'New Connection' dialog box with a title bar containing standard macOS window controls (red, yellow, green buttons). The main title is 'New Connection'. Below it, the section is titled 'Advanced Settings' with a subtitle 'Enter security information, if required by this cluster.' and a key icon. There are two main sections, each with a checkbox. The first section, 'This cluster requires credentials', has fields for 'Login:' and 'Password:'. The second section, 'This cluster requires SSL', has fields for 'Truststore file:' (with a browse button '...'), 'Truststore password:', 'Client authentication required' checkbox, 'Keystore file:' (with a browse button '...'), and 'Keystore password:'. At the bottom, there is a link 'Try to establish a connection' followed by the text 'using these settings.' and four buttons: '< Back', 'Next >', 'Cancel', and 'Finish'.

New Connection

Advanced Settings
Enter security information, if required by this cluster.

☐ This cluster requires credentials

Login:

Password:

☐ This cluster requires SSL

Truststore file: ...

Truststore password:

☐ Client authentication required

Keystore file: ...

Keystore password:

[Try to establish a connection](#) using these settings.

< Back Next > Cancel Finish

DevCenter uses a whitelist policy to connect with hosts in your cluster. DevCenter establishes a connection only to the nodes that have been added in the **Contact hosts** list in the **Connection** wizard.

LDAP support

LDAP authentication (in [DataStax Enterprise](#) clusters) is supported since version 1.5. DevCenter has been tested with the following LDAP providers:

- OpenLDAP
- OracleLDAP
- WinAD08
- WinAD12



Open connection

Open a connection either by:

- clicking the **Open connection** button in the toolbar
- Mac OS X: typing `#+F3`
- Windows, Linux: typing `Ctrl+F3`
- double-clicking the connection name



Close connection

Close a connection by:

- clicking the **Close connection** button in the toolbar
- Mac OS X: typing `#+F4`
- Windows, Linux: typing `Ctrl+F4`



Clone connection

Clone a connection by:

- clicking the **Clone connection** button in the toolbar
- Mac OS X: typing `#+#+D`
- Windows, Linux: typing `Ctrl+Shift+D`



Delete connection

Remove a connection by clicking

- the **Delete connection** button in the toolbar
- Mac OS X, Windows, Linux: typing `DEL`



Editing connection properties

Edit the properties of an existing connection by:

- right-clicking it and selecting **Properties**
- Mac OS X: typing `#+I`
- Windows, Linux: typing `Ctrl+I`

The **Properties for Connection** '*connection-name*' dialog displays.

Properties for Connection 'Dev Cassandra 2.1'

type filter text

Basic Settings

Enter information about the cluster to connect to.

Connection name: Dev Cassandra 2.1

Contact hosts: 127.0.0.2

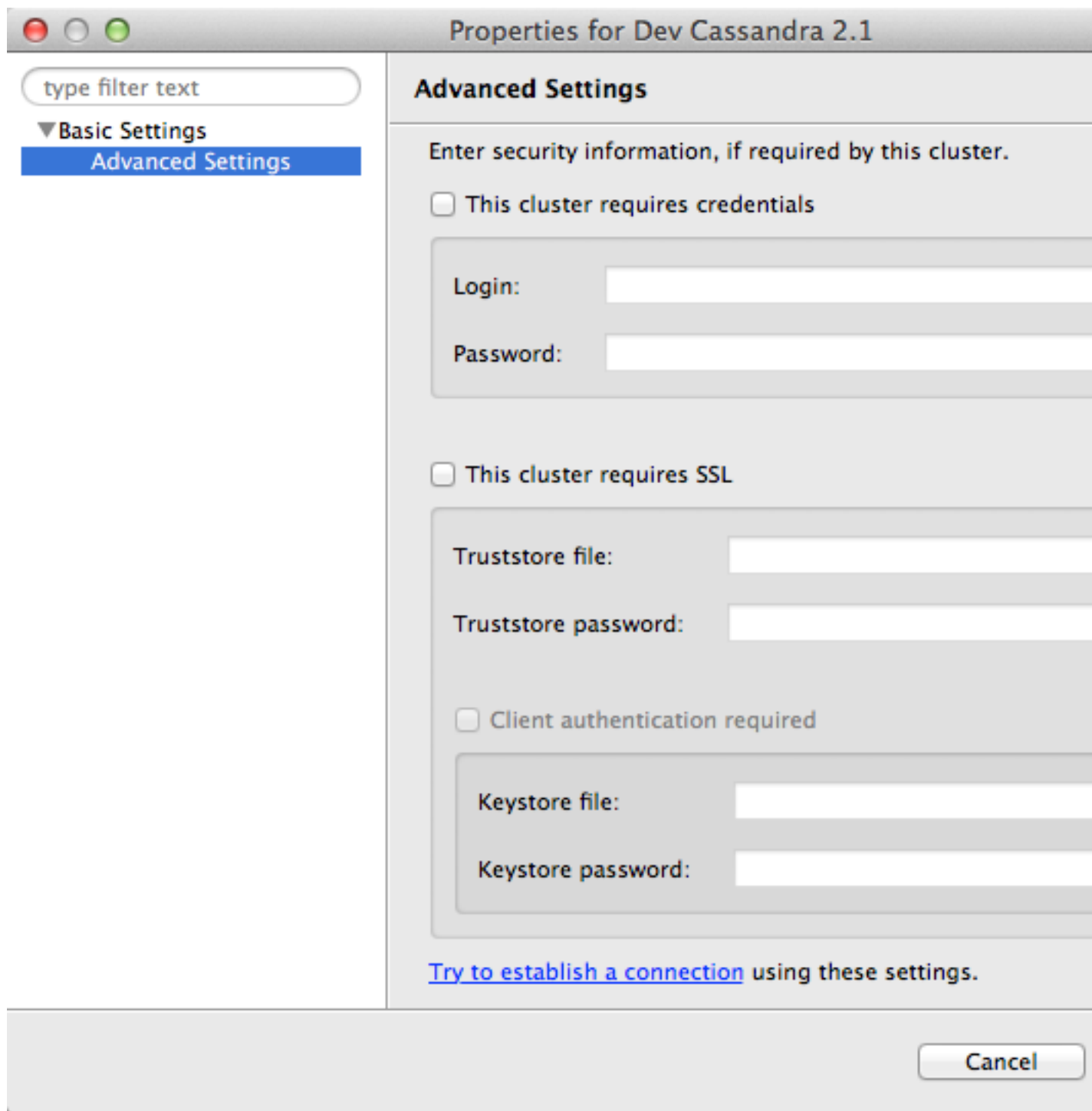
Native Protocol port: 9042

Use compression: ☒ None ☐ Snappy ☐ LZ4

[Try to establish a connection](#) using these settings.

Cancel

Edit the **Advanced Settings** by expanding the **Basic Settings** leaf.



The two panes of the **Properties** dialog are the same as in the [New connection](#) dialog.

Related reference

[Keyboard shortcuts](#) on page 66

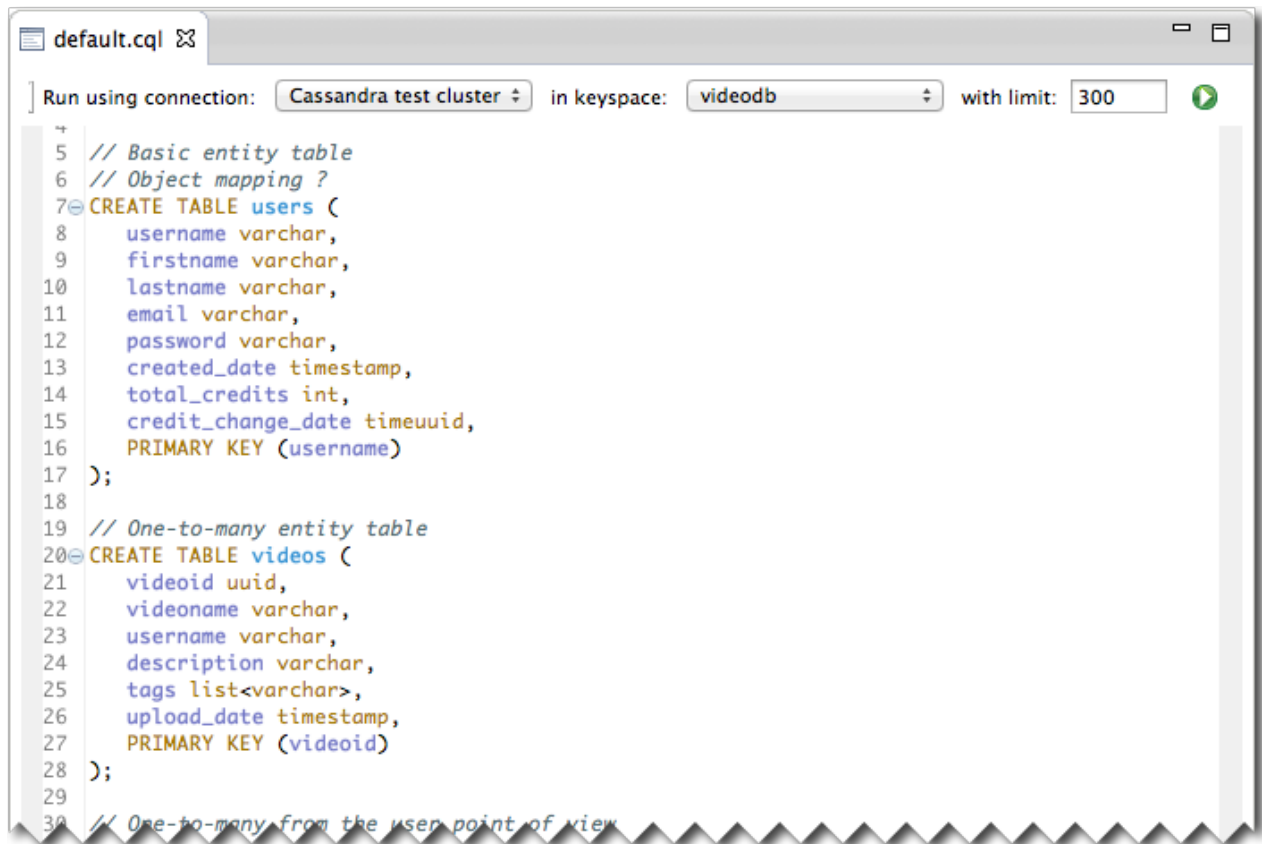
Query Editor

Editing

Create and edit a CQL script in a tab. Each tab is associated with a connection and the CQL script can be executed with a single click.

Besides normal editing functionality, such as cutting, copying, and pasting, the Query Editor also has:

- syntax highlighting for CQL
- code completion
- real-time error detection (errors are underlined in red, and hovering over the red error icon reveals the error)



Query editor contextual menu

Right-clicking on an element in the CQL displays a contextual menu. The **Source** submenu consists of:

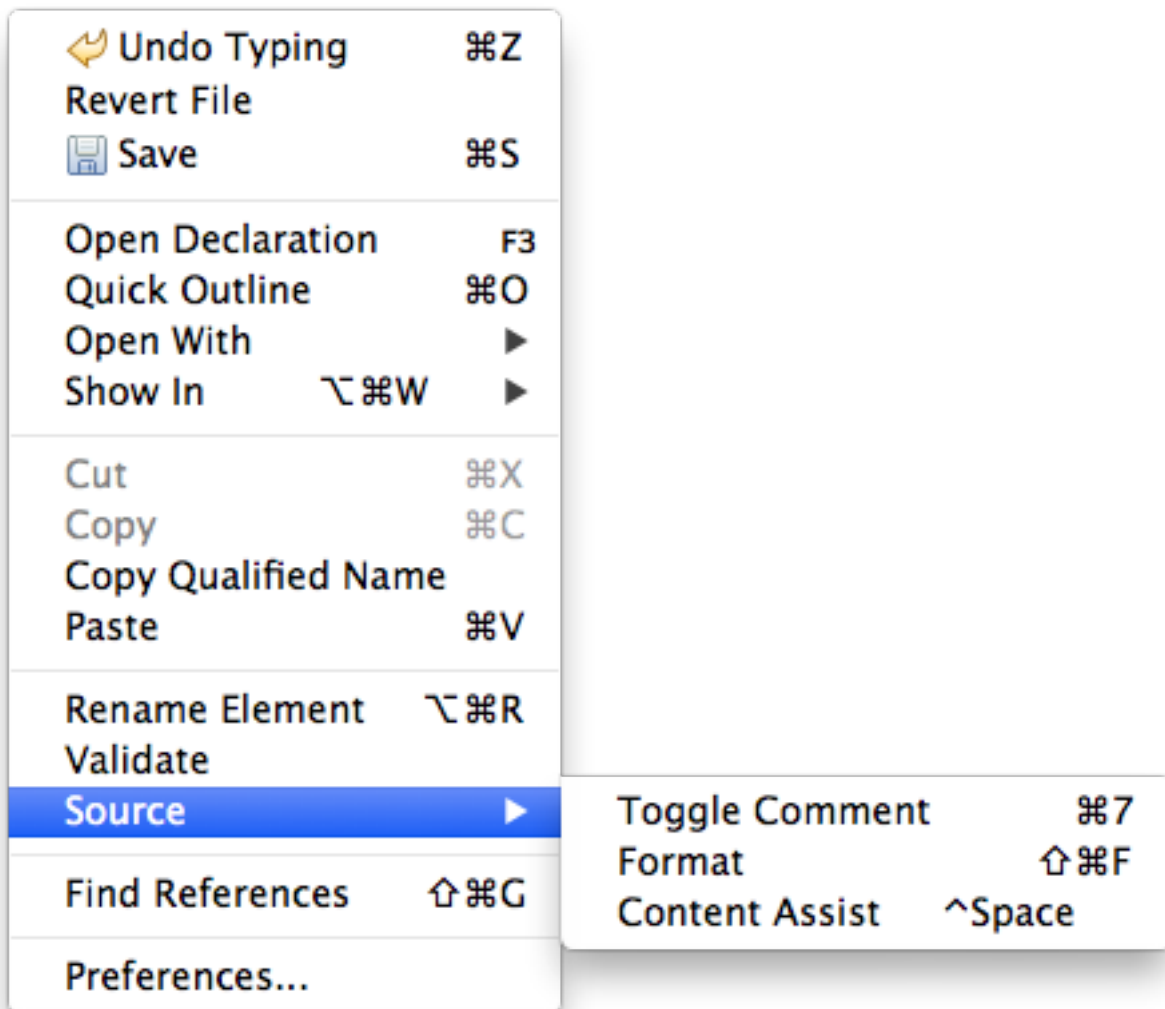
Source > Toggle Comment

toggles a comment on or off; works with single or multiple lines

Source > Format

formats CQL statement

Source > Content Assist



Run using connection

Run using connection: Cassandra test cluster in keyspace: videodb

Associate a tab in the Query Editor to:

- run with any of the connections in the **Connection Manager**
- use a chosen default keyspace
- limit the number of rows returned by a statement (limit 1000 rows returned)



Execute CQL Script

Execute all the CQL statements in the current editor tab.

By default, CQL statements are executed with a consistency level of `ONE`, but this can be changed to `QUORUM` or `ALL`. You can choose a different consistency level by clicking on the arrow to the right of the **Execute CQL Script** button and selecting the level.

Execute selected statement

Execute a single or multiple statements by highlighting them:

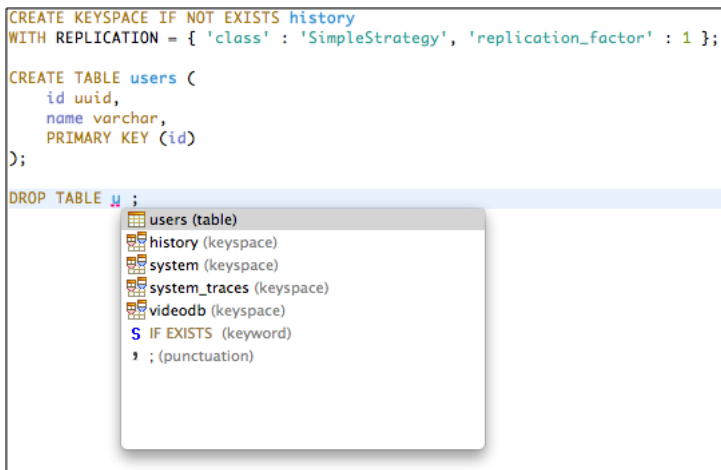
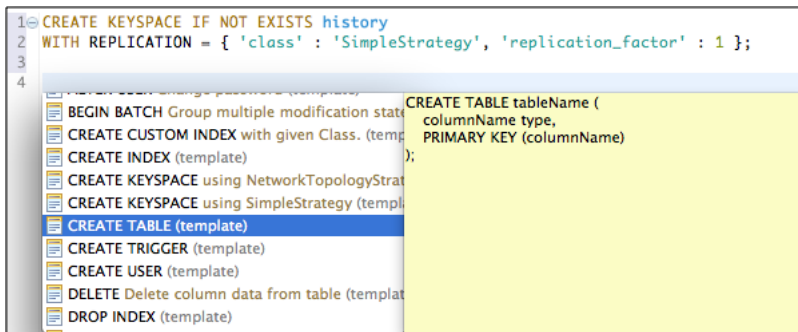
- Mac OS X: typing `#+F11`
- `Alt+F11` (Linux, Windows).

The statement may be partially highlighted also. When you execute a partially highlighted statement, the entire statement is first selected and then executed.

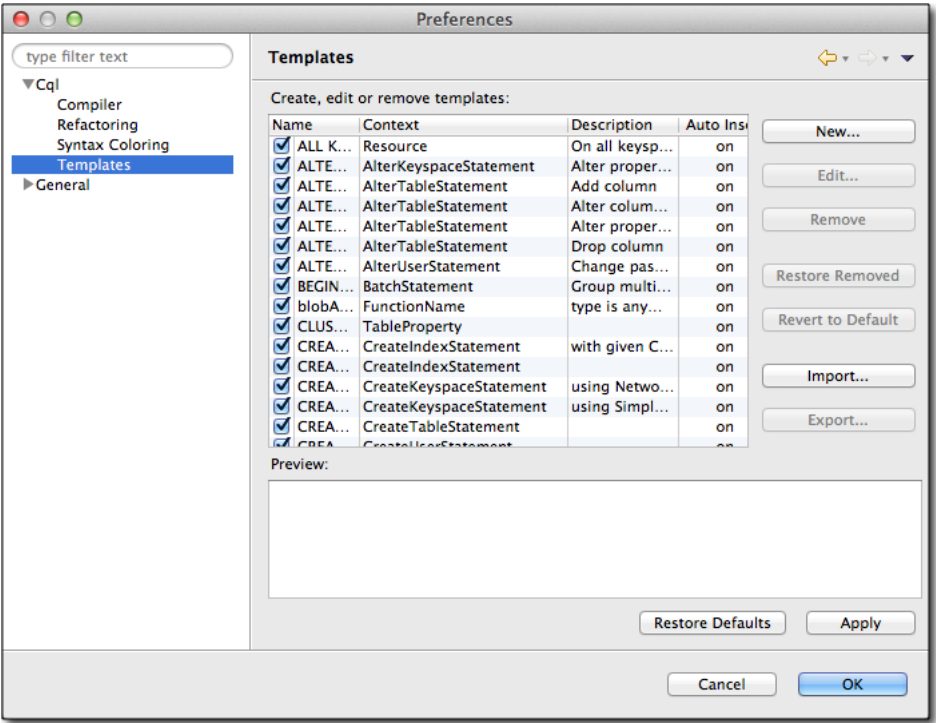
Code completion

The Query Editor has two types of code completion (activated by pressing **Ctrl-Spacebar**).

- code snippets: predefined complete CQL statements
- keywords and database identifiers: contextual code completion

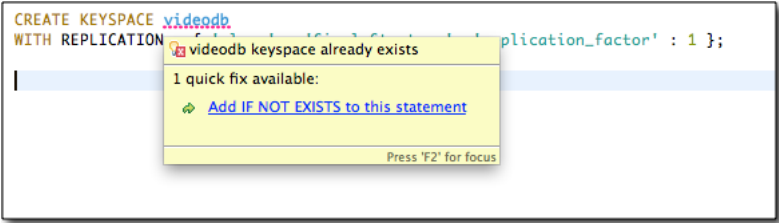


Add your own code snippets by using the **Preferences** dialog by selecting **File > Preferences > Cql > Templates**.



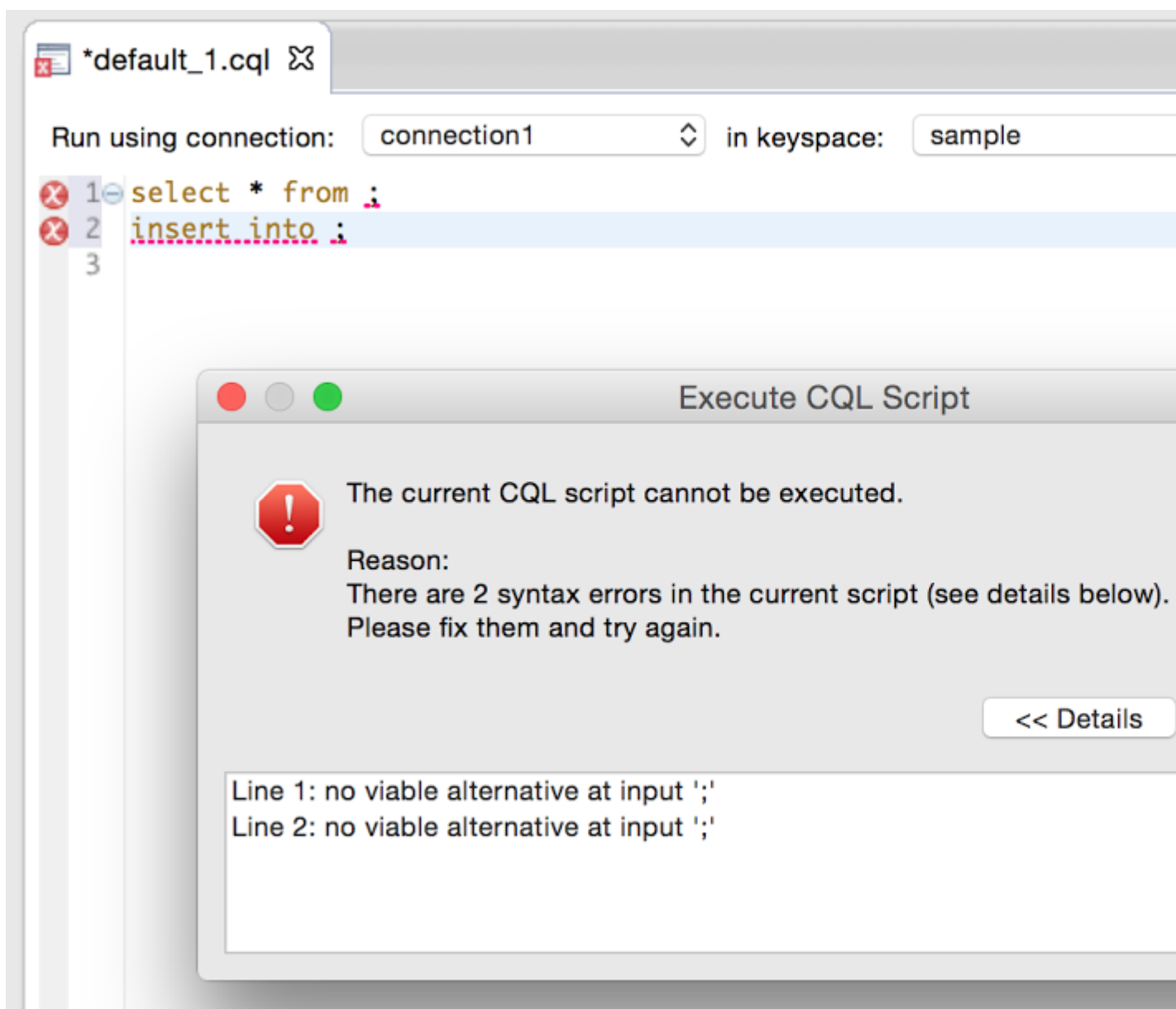
Real-time error detection and correction

Errors are detected by DevCenter in real-time (before a script is executed) based on the selected connection, keyspace, and other metadata about the cluster. The error is indicated by a red underline. By hovering the mouse cursor over the error a list containing possible quick fixes displays.



Preventing execution of CQL scripts containing syntax errors

If you try to execute CQL with a syntactic error in it, the code does not execute, and a dialog displays. This prevents DevCenter from sending incorrect or incomplete CQL statements. A syntax error disables the execution of the entire script.

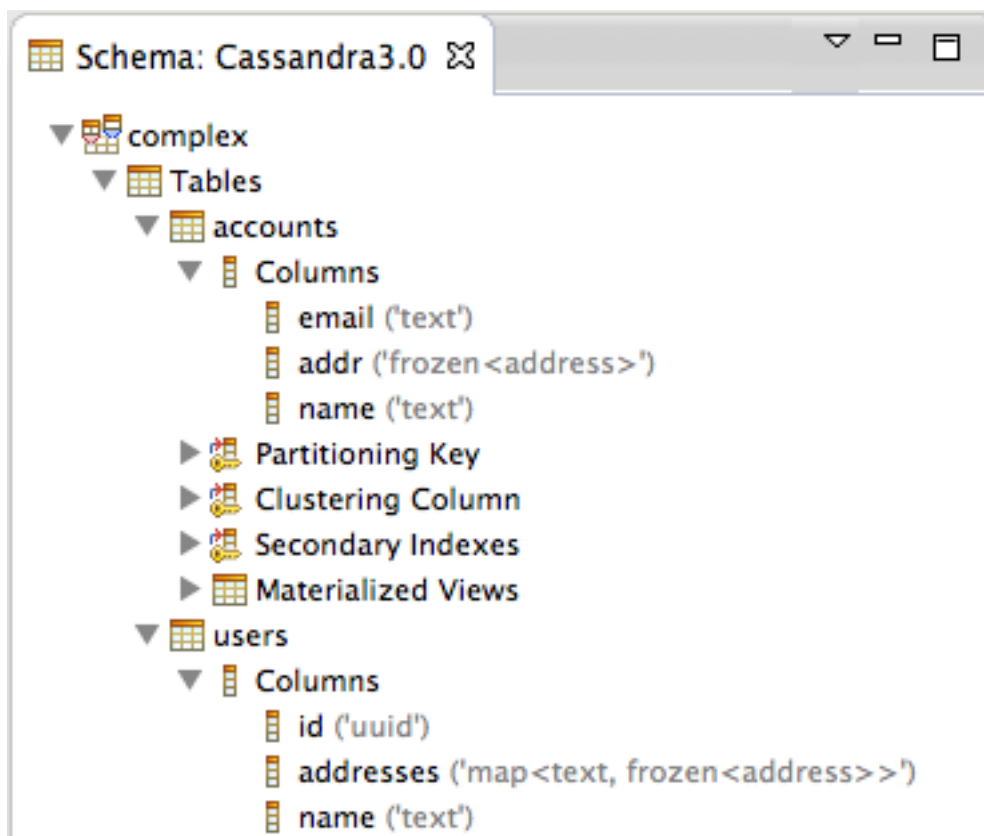


Schema Navigator

The **Schema Navigator** provides a tree control that displays the structure and details of the schema (for example, keyspaces, tables) for the current connection.

When DevCenter runs against a Cassandra cluster, the **Schema Navigator** displays User-defined functions, User-defined aggregates, and User-defined types (if version 1.4.1+) and **Materialized Views** (in version 1.5.0) which are not supported in Cassandra version 1.2.19. **Schema Navigator** only show nodes representing elements that are supported for the active Cassandra cluster.

It displays the schema for the current active tab in the **Query Editor** pane.



Schema navigator contextual menu

Right-clicking in the **Schema Navigator** pane (with no node selected) displays a contextual menu. The menu items are:

- **New Keyspace:** displays the **New Keyspace** wizard
- **New Table:** displays the **New Table** wizard

Keyspace node contextual menu

Right-clicking on a keyspace node in the **Schema Navigator** pane on a selected node displays a contextual menu. The menu items are:

New Keyspace

displays the **New Keyspace** wizard

New Table

displays the **New Table** wizard

New Index

displays the **New Index** wizard

New User Type

displays the **New User-defined Type** wizard

New Materialized View

displays the **New Materialized View** wizard

Clone Keyspace

displays the **New Keyspace** wizard with values of fields set to the selected keyspace's settings

Drop Keyspace

displays the **Drop Keyspace** dialog

Edit Keyspace

displays the **Edit Keyspace** wizard (which contains the same fields as the **New Keyspace** wizard) for editing the selected keyspace's settings

Note: If one of the system keyspace nodes (**system**, **system_auth**, **system_distributed**, etc.) is selected, then only the **New Keyspace** is available in the contextual menu, because it is not recommended to change this keyspaces.

Table node contextual menu

Right-clicking on a table node in the **Schema Navigator** pane on a selected node displays a contextual menu. The menu items are:

New Keyspace

displays the **New Keyspace** wizard

New Table

displays the **New Table** wizard

New Index

displays the **New Index** wizard

New User Type

displays the **New User-defined Type** wizard

New Materialized View

displays the **New Materialized View** wizard

Clone Table

displays the **New Table** wizard with values of fields set to the selected table's settings

Drop Table

displays the **Drop Table** dialog

Edit Table

displays the **Edit Table** wizard (which contains the same fields as the **New Table** wizard) for editing the selected table's settings

When editing a table, only primary key column names can be changed. Likewise, when editing a table's column, only non-primary key columns may have their types changed.

User-defined type node contextual menu

Right-clicking on a user-defined type node in the **Schema Navigator** pane on a selected node displays a contextual menu. The menu items are:

New Keyspace

displays the **New Keyspace** wizard

New Table

displays the **New Table** wizard

New Index

displays the **New Index** wizard

New User Type

displays the **New User-defined Type** wizard

New Materialized View

displays the **New Materialized View** wizard

Drop User Type

displays the **Drop User Type** dialog

Also, user-defined types can be dropped by selecting the node and selecting **Drop Function**

User-defined function node contextual menu

Right-clicking on a user-defined type node in the **Schema Navigator** pane on a selected node displays a contextual menu. The menu items are:

New Keyspace

displays the **New Keyspace** wizard

New Table

displays the **New Table** wizard

New Index

displays the **New Index** wizard

New User Type

displays the **New User-defined Type** wizard

New Materialized View

displays the **New Materialized View** wizard

Drop Function

displays the **Drop Function** dialog

Also, user-defined types can be dropped by selecting the node and selecting **Drop Function**

Materialized view node contextual menu

Right-clicking on a materialized view node in the **Schema Navigator** pane on a selected node displays a contextual menu. The menu items are:

New Keyspace

displays the **New Keyspace** wizard

New Table

displays the **New Table** wizard

New Index

displays the **New Index** wizard

New User Type

displays the **New User-defined Type** wizard

New Materialized View

displays the **New Materialized View** wizard

Clone View

displays the **New View** wizard with values of fields set to the selected table's settings

Drop View

displays the **Drop View** dialog

Edit View

displays the **Edit View** wizard (which contains the same fields as the **New View** wizard) for editing the selected table's settings

Wizards resynchronize with database after partial failure

If more than one statement is generated (typically in **Edit Keyspace** or **Edit Table** wizards), a statement may fail although previous ones succeeded. In this situation, the wizard dialog is kept open, and its contents are resynchronized with the database.

Keyspace wizards

The **New Keyspace** and the associated **Edit Keyspace** and **Clone Keyspace** wizards generate valid CQL statements that create, edit, or clone a new keyspace in the specified cluster.

New Keyspace wizard

To display the **New Keyspace** wizard:

- select the **File > New > Keyspace** menu item
- right-click in the schema navigator and select **New Keyspace**
- press the key shortcut `##++K`(Mac OS X) or `Ctrl+Alt+Shift+K` (Linux / Windows)

Navigate between panels using the **Previous** and **Next** buttons. A **Last** button also skips from the current panel to the **Summary** panel, if you won't be editing any fields on those panels.

The first panel (**Basic Settings**) of the **New Keyspace** wizard displays. There are two panes on this panel: one that displays keyspace settings and allows editing of them, and a **CQL Preview** pane that displays the CQL statement for creating the keyspace generated from the values in the first pane.

The editable fields are:

Connection

choose an existing connection or create a new one

Keyspace name

type in a valid keyspace name (invalid names are noted in red)

Replication strategy

which data replication strategy class (`SimpleStrategy` single data center or `NetworkTopologyStrategy` multiple data centers) to use; the strategy determines where replicas are placed

- **Replication factor:** which replication factor to use; the factor is the number of replicas across the cluster

Durable writes

by setting this property, data written to the keyspace bypasses the commit logs

New Keyspace

Basic Settings

Define the keyspace and properties. Click Next to review summary.

Connection:

Cassandra 2.1

New...

Keyspace name:

social

Replication Strategy:

SimpleStrategy

Replication factor:

3

☐ Durable writes:

☒

Reset properties above to their [default](#) values.

See more details about keyspace properties and their default values [here](#).

▼ CQL Preview:

```
CREATE KEYSPACE social
WITH replication = {
    'class' : 'SimpleStrategy',
    'replication_factor' : 3
};
```

Help

< Back

Next >

Cancel

Last >>

After selecting **Next**, the second panel (**Summary**) of the **New Keyspace** wizard displays.

The editable fields are:

- Whether to execute the generated statement(s) using the current connection or not
- Whether to insert the generated statement(s) into
 - a new CQL editor pane
 - an existing CQL editor pane

New Keyspace

Summary

Please review the generated statement(s) below and choose the appropriate actions to perform:

☒ Execute the generated statement(s) using connection Cassandra 2.1

☐ Insert the generated statement(s) into:

☒ a new CQL editor

☐ an existing CQL editor: default_1.cql

CQL Preview:

```
CREATE KEYSPACE social
WITH replication = {
    'class' : 'SimpleStrategy',
    'replication_factor' : 3
};
```

Help

< Back

Next >

Cancel

Finish

Both panels have an uneditable **CQL preview** pane showing the CQL statement to be generated according to the current values of the wizard's editable fields.

Edit Keyspace wizard

The **Edit Keyspace** and **Clone Keyspace** wizards have the same GUI panels as the **New Keyspace** wizard, but they load the constituent data in the fields so you can edit or clone a keyspace.

Note: If one of the system keyspace nodes (**system**, **system_auth**, **system_distributed**, etc.) is selected, then only the **New Keyspace** is available in the contextual menu, because it is not recommended to change this keyspaces.

Table wizards

The **New Table** and the associated **Edit Table** and **Clone Table** wizards generate valid CQL statements that create, edit, or clone a new UDT in the specified cluster.

New Table wizard

The **New Table** wizard consists of five panels:

- **Basic Settings**
- **Primary Key Settings**
- **Advanced Settings**
- **Compaction**
- **Summary**

Navigate between panels using the **Previous** and **Next** buttons. A **Last** button also skips from the current panel to the **Summary** panel, if you won't be editing any fields on those panels.

To display the **New Table** wizard:

- select the **File > New > Table** menu item
- right-click in the schema navigator and select **New Table**
- press the key shortcut **##++T**(Mac OS X) or **Ctrl+Alt+Shift+T** (Linux / Windows)

The first panel (**Basic Settings**) of the **New Table** wizard displays. There are four panes on this panel that display editable table settings and a **CQL Preview** pane that displays the CQL statement for creating the table generated from the values in the preceding three panes.

The editable fields are:

Connection

choose an existing connection or create a new one

Keyspace

choose an existing keyspace or create a new one

Table name

type in a valid table name (invalid names are noted in red)

Columns

add, remove, or move up or down a column

Comment

add or edit a comment property for the table

New Table

Basic Settings

Define the table's basic structure below. Click Next to fine-tune the table's primary key structure

Connection:

Cassandra 2.1

Keyspace:

social

Table name:

user

Columns:

Name	Type	Primary Key
id	timeuuid	<input checked="" type="checkbox"/>
userid	text	<input type="checkbox"/>
posts	int	<input type="checkbox"/>

Comment:

▼ CQL Preview:

```
CREATE TABLE social.user (  
  id timeuuid,  
  userid text,  
  posts int,  
  PRIMARY KEY (id)  
);
```

28

After selecting **Next**, the second panel (**Primary Key Settings**) of the **New Table** wizard displays.

The panel allows you to define the primary key structure for the table. They can be [simple or compound](#).

The editable fields are:

Available columns

displays the available columns

Partition keys

displays the partition key for the table (which determines the node on which the data is stored)

Clustering columns

displays the clustering columns for the table

New Table

Primary Key Settings

Define the table's primary key structure below. Click Next to set more advanced properties.

Available columns:

Partition keys:

id

Clustering columns:

Name	Order
userid	ASC
posts	DESC

▼ CQL Preview:

```

CREATE TABLE social.user (
  id timeuuid,
  userid text,
  posts int,
  PRIMARY KEY (id, userid, posts)
) WITH CLUSTERING ORDER BY ( userid ASC, posts DESC );

```

After selecting **Next**, the final panel (**Advanced Settings**) of the **New Table** wizard displays.

The editable fields are:

Table options

Compact storage

whether to store data so as to conserve disk space

Read repair chance

the basis for invoking read repairs on reads in clusters (a value between 0 and 1)

DC-local read repair chance

the probability of read repairs being invoked over all replicas in the current data center

Bloom filter FP chance

the desired false-positive probability for SSTable Bloom filters

Replicate on write

(Cassandra 2.0 only): if true, replicates writes to all affected replicas regardless of the consistency level specified by the client for a write request

Default time to live

the default expiration time in seconds for a table. Used in MapReduce scenarios when you have no control of TTL

GC grace seconds

specifies the time to wait before garbage collecting tombstones (deletion markers). The default value allows a great deal of time for consistency to be achieved prior to deletion. In many deployments this interval can be reduced, and in a single-node cluster it can be safely set to zero.

Minimum index interval

The minimum value to control the sampling of entries from the partition index, configure the sample frequency of the partition summary by changing these properties

Maximum index interval

The maximum value to control the sampling of entries from the partition index, configure the sample frequency of the partition summary by changing these properties

Index interval

(Cassandra 2.0 only): The interval value to control the sampling of entries from the partition index

Memtable flush period

Forces flushing of the memtable after the specified time in milliseconds elapses

Speculative retry

overrides normal read timeout when `read_repair_chance` is not 1.0, sending another request to read

Compression

SSTable compression

the compression to use (LZ4, Snappy, Default, or None)

Chunk length

size of block used by compression scheme

CRC check chance

the probability with which checksums are checked during read

Reset properties above to their **default** values

See more details above table properties and their default [here](#)

New Table

Advanced Settings

Define advanced properties for the table. Click Next to define compaction strategy.

Table options

☐ Compact storage
 ☒ Read repair chance: 0.1
 ☐ DC-local read repair chance: 0.1
 ☐ Bloom filter FP chance: 0.01
 ☐ Default time to live: 0
 ☐ GC grace seconds: 864000
 ☐ Minimum index interval: 128
 ☐ Maximum index interval: 2048
 ☐ Memtable flush period: 0
 ☐ Speculative retry: ☐ Always

Compression

☒ SSTable compression: ☒ LZ4 ☐ S
 ☐ Chunk length: 64
 ☐ CRC check chance: 1.0

Reset properties above to their [default](#) values.

See more details about table properties and their default values [here](#).

▼ CQL Preview:

```
CREATE TABLE social.users (
  id timeuuid,
  userid text,
  posts int,
  PRIMARY KEY (id, userid, posts)
) WITH CLUSTERING ORDER BY (userid ASC, posts DESC )
AND read_repair_chance = 0.1
AND compression = {
  'sstable_compression' : 'LZ4Compressor'
};
```


After selecting **Next**, the **Compaction** panel of the **New Table** wizard displays.

The editable fields are:

Strategy

sets the compaction strategy class to use

- **Enabled**
- **Tombstone threshold**
- **Tombstone compaction interval**
- **Unchecked tombstone compaction**
- **Min threshold**
- **Max threshold**
- **Min SSTable size**
- **Bucket low**
- **Bucket high**
- **Cold reads to omit**

Caching (Cassandra 2.1 and higher)

Keys

ALL or NONE

Rows per partition

number of CQL rows, NONE, or ALL

Caching (Cassandra 1.2 and 2.0)

Keys

ALL, KEYS_ONLY, ROWS_ONLY, or NONE

Reset properties above to their **default** values.

See more details above table properties and their default [here](#)

New Table

Compaction

Define the compaction strategy for the table. Click next to review summary.

☒ Strategy: SizeTieredCompactionStrategy

☒ Enabled:

☐ Tombstone threshold:

☐ Tombstone compaction interval:

☐ Unchecked tombstone compaction:

☐ Min threshold:

☐ Max threshold:

☐ Min SSTable size:

☐ Bucket low:

☐ Bucket high:

☐ Cold reads to omit:

☒

0.2

↑

↓

86400

↑

↓

☐

4

↑

↓

32

↑

↓

50

↑

↓

0.5

↑

↓

1.5

↑

↓

0.05

↑

↓

Caching

☐ Keys:

☐ Rows per partition:

☒ All ☐ No

☐ All ☒ No

Reset properties above to their [default](#) values.

See more details about table properties and their default values [here](#).

▼ CQL Preview:

```

CREATE TABLE social.users (
  id timeuuid,
  userid text,
  posts int,
  PRIMARY KEY (id, userid, posts)
) WITH CLUSTERING ORDER BY ( userid ASC, posts DESC )
AND read_repair_chance = 0.1
AND compression = {
  'sstable_compression' : 'LZ4Compressor'
}
AND compaction = {
  'class' : 'SizeTieredCompactionStrategy',

```

After selecting **Next**, the final **Summary** panel of the **New Table** wizard displays.

The editable fields are:

- Execute the generated statement(s) using connection Cassandra 3.0
- Insert the generated statements into:
 - a new CQL editor
 - an existing CQL editor

New Table

Summary

Please review the generated statement(s) below and choose the appropriate actions to perform:

☒ Execute the generated statement(s) using connection Cassandra 2.1

☐ Insert the generated statement(s) into:

☒ a new CQL editor

☐ an existing CQL editor:

two_one_features.cql

▼ CQL Preview:

CREATE TABLE social.users (
 id timeuuid,
 userid text,
 posts text,
 PRIMARY KEY (id, userid, posts)
) WITH CLUSTERING ORDER BY (userid ASC, posts DESC);

36

Table wizards

The **Edit Table** and **Clone Table** wizards have the same GUI panels as the **New Table** wizard, but they load the constituent data in the fields so you can edit or clone a table.

Index wizard wizards

The **New Index** and the associated **Edit Index** and **Clone Index** wizards generate valid CQL statements that create, edit, or clone a new UDT in the specified cluster.

New User-defined Type wizard

The **New Index** wizard provides a visual way to generate valid CQL statements that create one or more secondary indices for the specified table.

To display the **New Index** wizard:

1. Select the **File > New > Index** menu item
2. Right-click in the schema navigator and select **New Index**
3. Press the key shortcut `###+I` (Mac OS X) or `Ctrl+Alt+Shift+I` (Linux / Windows)

Navigate between panels using the **Previous** and **Next** buttons. A **Last** button also skips from the current panel to the **Summary** panel, if you won't be editing any fields on those panels.

The first panel (**Create Index**) of the **Create Index** wizard displays. There are two panes on this panel: one that displays index settings and allows editing of them, and a **CQL Preview** pane that displays the CQL statement(s) for creating the index generated from the values in the first pane.

The editable fields are:

Connection

choose an existing connection or create a new one

Keyspace

choose an existing keyspace or create a new one

Table

choose an existing table or create a new one

Secondary indices

add valid index names and columns

○ ○ ○

Create index

Create index

Define new indices. Click Next to review summary.

Connection:

Cassandra 2.1

New...

Keyspace:

social

New...

Table:

users

New...

Secondary indices:

Index name	Column
user_address	address

Add

Remove

▼ CQL Preview:

```
CREATE INDEX user_address ON social.users (address);
```

Help

< Back

Next >

Cancel

Last >>

After selecting **Next**, the second panel (**Summary**) of the **New Index** wizard displays.

In the **Summary** panel you can review the final CQL that will be generated and choose between the following actions:

- Whether to execute the generated statement(s) using the current connection or not
- Whether to insert the generated statement(s) into
 - a new CQL editor pane
 - an existing CQL editor pane

○ ○ ○

Create index

Summary

Please review the generated statement(s) below and choose the appropriate actions to perform:

☒ Execute the generated statement(s) using connection Cassandra 2.1

☐ Insert the generated statement(s) into:

☒ a new CQL editor

☐ an existing CQL editor:

two_one_features.cql ▾

▼ CQL Preview:

CREATE INDEX user_address ON social.users (address);

Help

< Back

Next >

Cancel

Finish

Both panels have an uneditable **CQL preview** pane showing the CQL statement to be generated according to the current values of the wizard's editable fields.

Edit and Clone Index wizards

The **New Index** and **Clone Index** wizards have the same GUI panels as the **New Index** wizard, but they load the constituent data in the fields so you can edit or clone an index.

User-defined Type wizards

The **New User-defined Type** and the associated **Edit User-defined Type** and **Clone User-defined Type** wizards generate valid CQL statements that create, edit, or clone a new UDT in the specified cluster.

New User-defined Type wizard

To display the **New User-defined Type** wizard:

- select the **File > New > User-defined Type** menu item
- right-click in the schema navigator and select **New User-defined Type**
- press the key shortcut **##++U** (Mac OS X) or **Ctrl+Alt+Shift+U** (Linux / Windows)

Navigate between panels using the **Previous** and **Next** buttons. A **Last** button also skips from the current panel to the **Summary** panel, if you won't be editing any fields on those panels.

The first panel (**Basic Settings**) of the **New User-defined Type** wizard displays. There are two panes on this panel: one that displays UDT settings and allows editing of them, and a **CQL Preview** pane that displays the CQL statement for creating the UDT generated from the values in the first pane.

In the **Summary** panel you can review the final CQL that will be generated and choose between the following actions:

Connection

choose an existing connection or create a new one

Keyspace

choose an existing keyspace or create a new one

Type name

type a valid type name

Fields

add valid type names and types for the UDT's fields

New User-defined Type

Basic Settings

Define the user-defined type and properties. Click Next to review summary.

Connection:

Cassandra 2.1

New...

Keyspace:

social

New...

Type name:

post

Fields:

Name	Type
user	text
body	text

Add

Remove

Up

Down

▼ CQL Preview:

CREATE TYPE social.post (
 user text,
 body text
);

Help

< Back

Next >

Cancel

Last >>

After selecting **Next**, the second panel (**Summary**) of the **User-defined Type** wizard displays.

In the **Summary** panel you can review the final CQL that will be generated and choose between the following actions:

- Whether to execute the generated statement(s) using the current connection or not
- Whether to insert the generated statement(s) into
 - a new CQL editor pane
 - an existing CQL editor pane

New User-defined Type

Summary

Please review the generated statement(s) below and choose the appropriate actions to perform:

☒ Execute the generated statement(s) using connection Cassandra 2.1

☐ Insert the generated statement(s) into:

☒ a new CQL editor

☐ an existing CQL editor:

two_one_features.cql

▼ CQL Preview:

CREATE TYPE social.post (
 user text,
 body text
);

Help

< Back

Next >

Cancel

Finish

Both panels have an uneditable **CQL preview** pane showing the CQL statement to be generated according to the current values of the wizard's editable fields.

Edit and Clone User Type wizards

The **Edit User Type** and **Clone User Type** wizards have the same GUI panels as the **New User Type** wizard, but they load the constituent data in the fields so you can edit or clone a UDT.

Materialized View wizards

The **New Materialized View** wizards generate valid CQL statements that create, edit, or clone a new materialized view in the specified cluster.

New Materialized View wizard

The **New Materialized View** wizard consists of five panels:

- **Basic Settings**
- **Primary Key Settings**
- **Advanced Settings**
- **Compaction**
- **Summary**

To display the **New Materialized View** wizard:

- select the **File > New > Materialized View** menu item
- right-click in the schema navigator and select **New Materialized View**
- press the key shortcut `##++v` (Mac OS X) or `Ctrl+Alt+Shift+v` (Linux / Windows)

Navigate between panels using the **Previous** and **Next** buttons. A **Last** button also skips from the current panel to the **Summary** panel, if you won't be editing any fields on those panels.

The first panel (**Basic Settings**) of the **New Materialized View** wizard displays. There are two panes on this panel: one that displays materialized view settings and allows editing of them, and a **CQL Preview** pane that displays the CQL statement for creating the materialized view generated from the values in the first pane.

In the **Basic Settings** panel, the editable fields are:

Connection

choose an existing connection or create a new one

Keyspace

choose an existing keyspace or create a new one

Table

choose an existing table or create a new one

View name

type a valid materialized view name

Available columns

select an available column

New Materialized View

Basic Settings

Define the view's basic structure below.

Connection:

Cassandra3.0

Keyspace:

complex

Table:

accounts

View name:

RoomWitha

Available columns:

Selected

name

email

>

<

▼ CQL Preview:

```
CREATE MATERIALIZED VIEW complex."RoomWitha" AS
  SELECT email
  FROM accounts
  WHERE email IS NOT NULL AND addr IS NOT NULL
  PRIMARY KEY ( email, addr );
```

After selecting **Next**, the second panel (**Primary Key Settings**) of the **New Materialized View** wizard displays.

In the **Primary Key Settings** panel you can define the materialized view's primary key structure. The editable fields are:

Available columns

displays the available columns

Partition keys

displays the partition key for the table (which determines the node on which the data is stored)

Clustering columns

displays the clustering columns for the table

[illegible]

After selecting **Next**, the final panel (**Advanced Settings**) of the **New Materialized View** wizard displays.

The editable fields are:

Materialized View options

Read repair chance

the basis for invoking read repairs on reads in clusters (a value between 0 and 1)

DC-local read repair chance

the probability of read repairs being invoked over all replicas in the current data center

Bloom filter FP chance

the desired false-positive probability for SSTable Bloom filters

Default time to live

the default expiration time in seconds for a table. Used in MapReduce scenarios when you have no control of TTL

GC grace seconds

specifies the time to wait before garbage collecting tombstones (deletion markers). The default value allows a great deal of time for consistency to be achieved prior to deletion. In many deployments this interval can be reduced, and in a single-node cluster it can be safely set to zero.

Minimum index interval

The minimum value to control the sampling of entries from the partition index, configure the sample frequency of the partition summary by changing these properties

Maximum index interval

The maximum value to control the sampling of entries from the partition index, configure the sample frequency of the partition summary by changing these properties

Index interval

(Cassandra 2.0 only): The interval value to control the sampling of entries from the partition index

Memtable flush period

Forces flushing of the memtable after the specified time in milliseconds elapses

Speculative retry

overrides normal read timeout when `read_repair_chance` is not 1.0, sending another request to read

CRC check chance

the probability with which checksums are checked during read

Compression

Enabled

whether compression is enabled or not

SSTable compression

the compression to use (LZ4, Snappy, Default, or None)

Chunk length

size of block used by compression scheme

Reset properties above to their **default** values

See more details above table properties and their default [here](#)

New Materialized View

Advanced Settings

Define advanced properties for the view. Click Next to define compaction strategy.

Materialized View options

☐ Read repair chance: 0.0
 ☐ DC-local read repair chance: 0.1
 ☐ Bloom filter FP chance: 0.01
 ☐ Default time to live: 0
 ☐ GC grace seconds: 864000
 ☐ Minimum index interval: 128
 ☐ Maximum index interval: 2048
 ☐ Memtable flush period: 0
 ☐ Speculative retry: Always
 ☐ CRC check chance: 1.0

Compression

☐ Enabled: ☒
☐ SSTable compression: LZ4
 ☐ Chunk length: 64

Reset properties above to their [default](#) values.

See more details about view properties and their default values [here](#).

▼ CQL Preview:

```
CREATE MATERIALIZED VIEW complex."RoomWitha" AS
SELECT email
FROM accounts
WHERE email IS NOT NULL AND addr IS NOT NULL
PRIMARY KEY ( email, addr );
```

After selecting **Next**, the **Compaction** panel of the **New Materialized View** wizard displays.

The editable fields are:

Strategy

sets the compaction strategy class to use

- **Enabled**
- **Tombstone threshold**
- **Tombstone compaction interval**
- **Unchecked tombstone compaction**
- **Min threshold**
- **Max threshold**
- **Min SSTable size**
- **Bucket low**
- **Bucket high**

Caching (Cassandra 2.1 and higher)

Keys

ALL or NONE

Rows per partition

number of CQL rows, NONE, or ALL

Caching (Cassandra 1.2 and 2.0)

Keys

ALL, KEYS_ONLY, ROWS_ONLY, or NONE

Reset properties above to their **default** values.

See more details above table properties and their default [here](#)

New Materialized View

Compaction

Define the compaction strategy for the view. Click next to review summary.

☐ Strategy: SizeTieredCompactionStrategy

☐ Enabled: ☒
☐ Tombstone threshold: 0.2
☐ Tombstone compaction interval: 86400
☐ Unchecked tombstone compaction: ☐
☐ Min threshold: 4
☐ Max threshold: 32
☐ Min SSTable size: 50
☐ Bucket low: 0.5
☐ Bucket high: 1.5

Caching

☐ Keys: ☒ All ☐ No
☐ Rows per partition: ☐ All ☒ No

Reset properties above to their [default](#) values.

See more details about view properties and their default values [here](#).

▼ CQL Preview:

```
CREATE MATERIALIZED VIEW complex."RoomWitha" AS
SELECT email
FROM accounts
WHERE email IS NOT NULL AND addr IS NOT NULL
PRIMARY KEY ( email, addr );
```

After selecting **Next**, the final **Summary** panel of the **New Materialized View** wizard displays.

The editable fields are:

- Execute the generated statement(s) using connection Cassandra 3.0
- Insert the generated statements into:
 - a new CQL editor
 - an existing CQL editor

New Materialized View

Summary

Please review the generated statement(s) below and choose the appropriate actions to perform:

☒ Execute the generated statement(s) using connection Cassandra3.0

☐ Insert the generated statement(s) into:

- ☒ a new CQL editor
- ☐ an existing CQL editor:

▼ CQL Preview:

```
CREATE MATERIALIZED VIEW complex."RoomWitha" AS
  SELECT email
  FROM accounts
  WHERE email IS NOT NULL AND addr IS NOT NULL
  PRIMARY KEY ( email, addr );
```

Edit View wizard

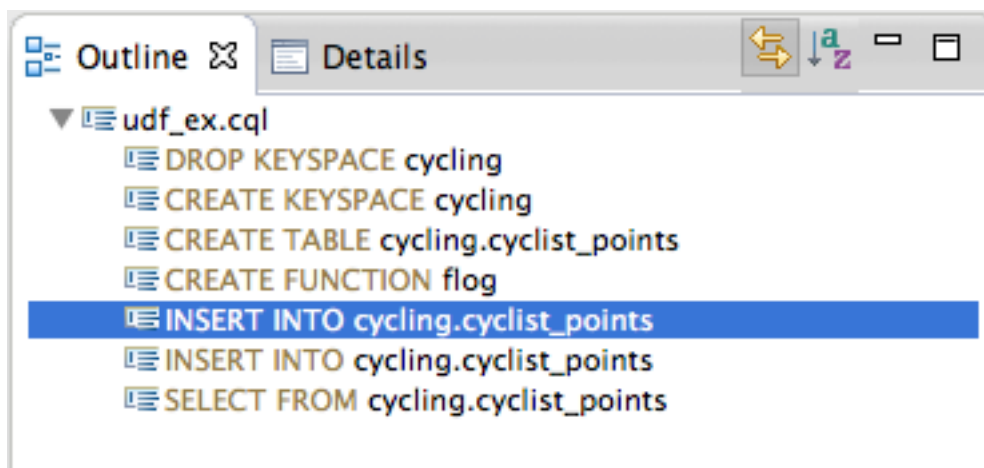
The **Edit View** and **Clone View** wizards have fewer steps because some of the details of an already defined materialized view cannot be changed. You access the **Edit View** and **Clone View** wizards by right-clicking on an existing materialized view node in the **Schema Navigator** and selecting the respective menu item in the contextual menu.

The **Edit View** wizard starts out with the **Advanced Settings** and includes the **Compaction** and **Summary** panels because you cannot edit the values on the **Basic Settings** and **Primary Key Settings** panels.

On the other hand, the **Clone View** wizard does start with the **Basic Settings** panel and loads the constituent data from the cloned view into its panels.

Outline

The **Outline** tab allows for quick navigation within large CQL scripts by selecting the statement in the tree that section of the **Query Editor** where the statement occurs is displayed.



Link to Editor

Toggles between whether the CQL statement is linked to its location in the Query Editor tab or not.



Order Alphabetically

Toggles between whether the CQL statements are listed in alphabetical order or in the order which they occur in the CQL script in.

Results

The **Results** pane has two parts:

- a tabular grid which displays results from the last query executed
- a status bar which displays pertinent information about executed statements

Results	Query Trace
id	addresses
93031620-12a...	{home={street:'123 Eddy St.', city:'Petaluma', zip_code:95566, phones:[{alias:'ho
756716f7-2e5...	{home={street:'1021 West 4th St. #202', city:'San Pedro', zip_code:92330, phone
f6071e72-48ec...	{home={street:'2580 Arnold Dr.', city:'San Fransisco', zip_code:94110, phones:[

In the grid display, you can:

- reorder and resize columns
- sort rows by a column by clicking on the column heading
- copy the data from the results by
 - a single selected row (click on the row)
 - multiple rows **#**+click (Macintosh) / **Ctrl**+click (Windows / Linux)
 - all of the rows (either as CSV or an INSERT statement)

Details tab

If you select a cell in the **Results** tab, you can view more information about the column and its value in the **Details** data viewer tab.

The **Details** data viewer tab (in the **Outline** view) displays information about the currently selected cell in the **Results** pane. It displays information about the type, the complete value, and also the TTL and timestamp of the selected cell.

1 select * from videos;

Results

Query Trace

videoid	description	location	tags	upload_date	username
99051fe9-6a9c-...	My cat likes to...	{US=/us/vid/99/...	{cats, lol, piano}	Fri Jun 01 08:00...	tcodd
873ff430-9c23-...	Second in a thr...	{YouTube=http:...	{cassandra, cql...	Thu May 16 16:...	pmcfadin
0c3f7e87-f6b6-...	An overview of...	{US=/us/vid/0c...	{book, databas...	Mon Sep 03 10:...	cdate
416a5ddc-00a5...	I think there mi...	{US=/us/vid/41...	{brewer, cap, d...	Sat Dec 01 11:...	cdate
06049cbb-dfed...	First in a three...	{YouTube=http:...	{cassandra, dat...	Thu May 02 12:...	pmcfadin
49f64d40-7d89...	Third in a three...	{YouTube=http:...	{cassandra, dat...	Tue Jun 11 11:...	pmcfadin
b3a76c6b-7c7f...	My dog learned...	{US=/us/vid/b3...	{dogs, lol, piano}	Thu Aug 30 16:...	tcodd

datatypes

keyspace1

...

Details

Type: set<varchar>



Timestamp: [Not av

TTL: [Not available]

Value:

["cats", "

Query Trace

Results	Query Trace	
Trace session id: a5272a30-5084-11e4-826e-b9640cbdc5dc [more]		
Node information  127.0.0.2  127.0.0.3		Role information C – Coordinator node R – Replica node CR – Coordinator and replica
Activity	Timestamp	Role
Execute CQL3 query	06:52:16.067000	C
Parsing SELECT * from simplex.songs LIMIT 300	06:52:16.067000	C
Preparing statement	06:52:16.068000	C
Computing ranges to query	06:52:16.068000	CR
Submitting range requests on 3 ranges with a concurrency of 1 (0....	06:52:16.068000	CR
Enqueuing request to /127.0.0.3	06:52:16.068000	C
Submitted 1 concurrent range requests covering 3 ranges	06:52:16.068000	CR
Sending message to /127.0.0.3	06:52:16.068000	C

See also:

- [Tracing](#)
- [Request tracing in Cassandra 1.2](#)
- [Advanced request tracing](#)

Trace session_id

The Trace `session_id` uniquely identifies the tracing session. Trace data is saved in Cassandra for 24 hours and can be queried using statements found by clicking the 'more' link. For example,

```
Tracing session information is stored for 24 hours. Session and session
event data can be queried with:
SELECT * FROM system_traces.sessions WHERE session_id =
d6aa92a0-4f48-11e4-8d9b-65dde8fcb188;
SELECT * FROM system_traces.events WHERE session_id =
d6aa92a0-4f48-11e4-8d9b-65dde8fcb188;
```

Node information

The **Node information** section lists each node participating in the query along with a color coding for easier identification in the trace table results.

Role information

The **Role Information** section describes the roles that participating nodes may play in each trace event:

C

Coordinator node

R

Replica node

CR

Coordinator and Replica node

Trace columns

The columns in the **Query trace** are:

Activity

Description of the activity in the trace event.

Timestamp

The timestamp of the event on the source node.

Role

The role of the node for this event (Coordinator, Replica, Coordinator and Replica).

Source node

The source node where the event occurred.

Source elapsed (μ s)

Elapsed time of the event on the source node in microseconds.

Thread

The name of the thread on the source node that executed the event.

Disabling or enabling Query Trace

Query Trace is enabled in DevCenter by default. Starting in version 1.5, you can disable Query Trace by adding or editing a property to the `DevCenter.ini` file.

```
devcenter.querytrace.enabled=false
```

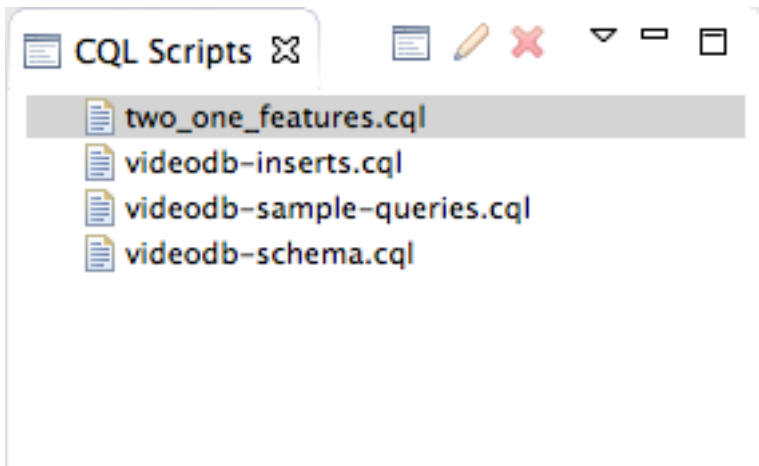
The location of the `DevCenter.ini` file:

- Linux: `DevCenter_Home/DevCenter.app/Contents/MacOS`
- Mac OS X: `DevCenter_Home`
- Windows: `DevCenter_Home`

CQL Scripts

There are two types of files: scratchpads (new scripts having a default name) and already saved scripts. DevCenter saves CQL scripts (both scratchpads and named scripts) in your DevCenter workspace under these directories:

- scratchpads: `user-home / .devcenter/DevCenter/.default`
- CQL scripts: `user-home / .devcenter/DevCenter/CQLScripts`



Select one or more CQL scripts by typing **#+click** (Mac OS X) / **Ctrl+Click** (Windows / Linux).

Note: When you open a CQL script for editing using **File > Open File ...**, any changes you make are to your workspace copy of the CQL script and not to the original file.



New CQL script

Create a new CQL script by selecting the **File > New CQL Script**, typing **#+#+N** (Mac OS X) / **Ctrl+Shift+N** (Windows, Linux), or clicking the **New CQL Script** icon in the toolbar of this view.



Open CQL script

Edit an existing script by clicking the **Open CQL script** action or typing **#+F3** (Mac OS X) / **Ctrl+F3** (Windows, Linux).

Also import an existing script by selecting **File > Open File**.



Delete CQL script

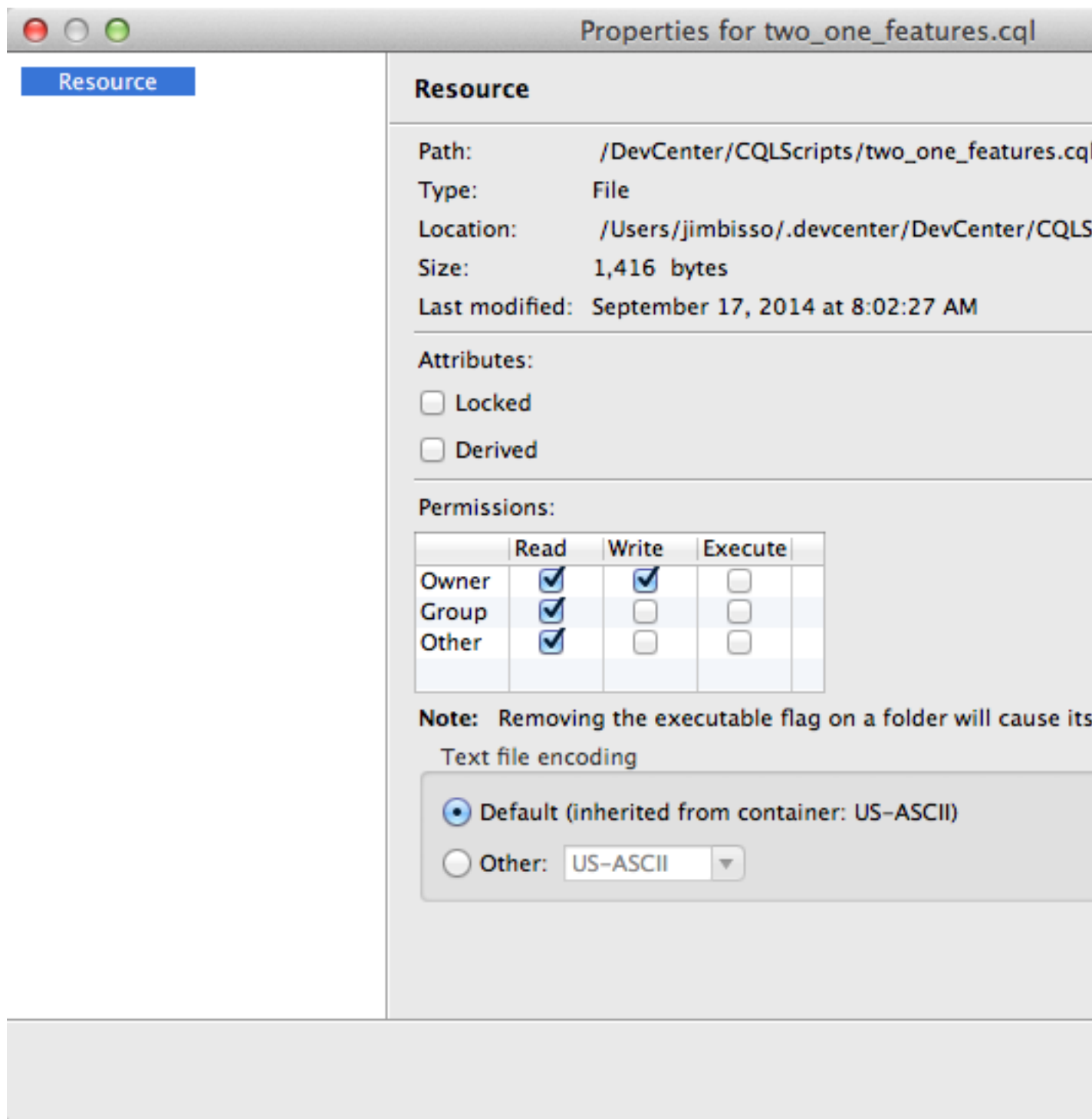
You delete a script by:

- clicking the **Delete CQL script** icon
- Mac OS X, Windows, Linux: typing **DEL**

Note: In case you must delete your DevCenter workspace directory, make sure you have copied or moved the existing CQL scripts to a different location on your disk.

Properties

You view a CQL script's properties by right-clicking on the script and selecting **Properties** in the contextual menu or typing **#+I** (Macintosh) / **Ctrl+I** (Windows, Linux).



You see all the file information (for example, the time the file was last modified and its read, write, execute permissions).

Related reference

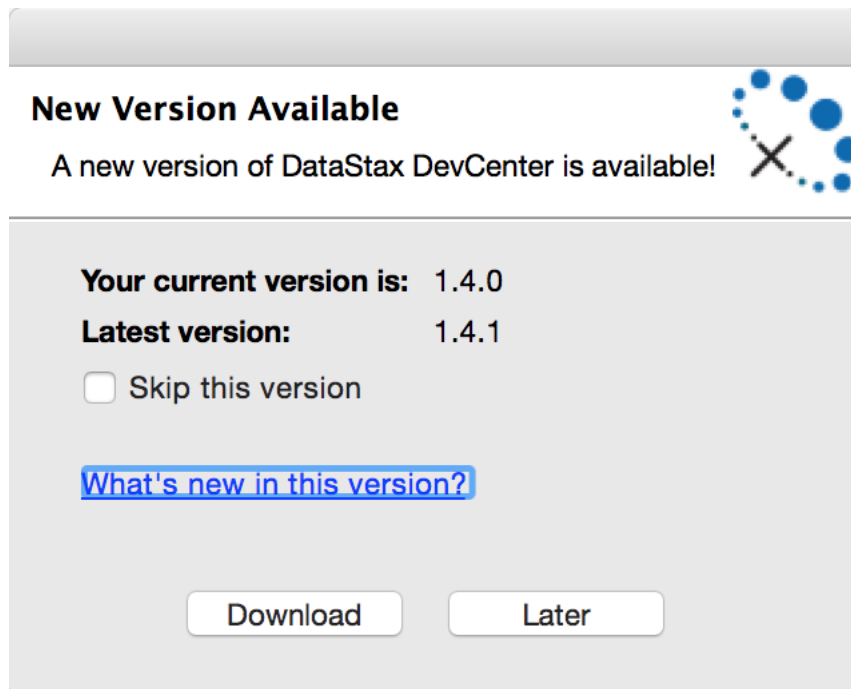
[Keyboard shortcuts](#) on page 66

Automatic updates

Starting with version 1.4, DevCenter can notify you when a new version is available. After installing DevCenter 1.4 you will be prompted to configure this option. You can also configure at any time by accessing **Preferences > Update**.

When an update is available a dialog is displayed that includes information about the currently installed version and the new version available.

You can choose to downloading it now or a later time or skip this version altogether.



Submit Feedback form

By selecting **Help > Submit Feedback**, you can send feedback directly to DataStax. The only required field is the **Comment**. The submitted feedback includes installation details for the computer you are running DevCenter on.

Submit DevCenter Feedback

Use this form to share your comments or ask a question. We'd love to hear from you!

Comments (required):

Name:

Email:

If you provide your email we will be able to follow up with you.

Note: The submitted form will include installation details so we can further investigate any reported issues. To view the included information, please click the Installation Details button and then select the Configuration tab.

Working with JSON data

Starting with version 2.2 of Cassandra, data can be INSERTed and SELECTed into tables with the data represented as JSON objects. The members of the object map to the column names in the table. For example:

With this schema:

```
CREATE TABLE IF NOT EXISTS social.users
(id uuid PRIMARY KEY, name text);
```

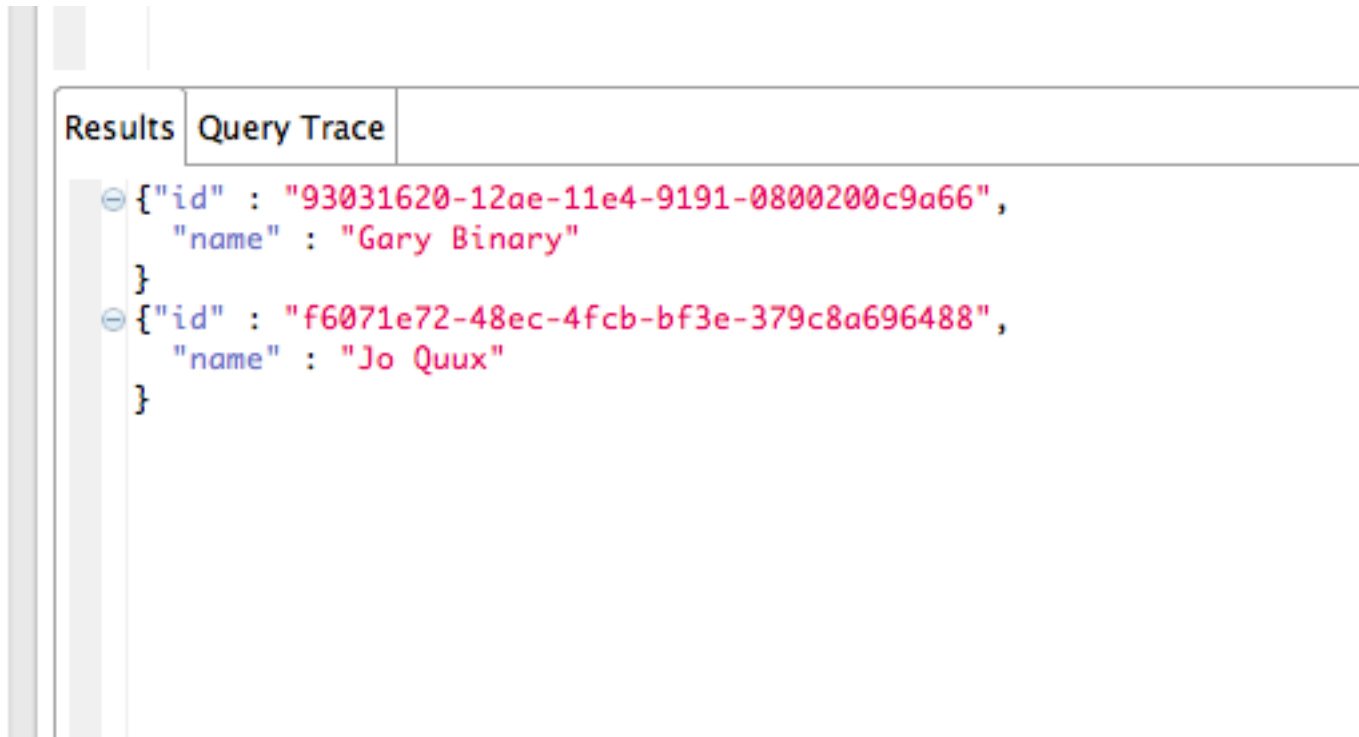
And this data:

```
INSERT INTO social.users JSON '{ "id" :
  "93031620-12ae-11e4-9191-0800200c9a66", "name" : "Gary Binary"}';

INSERT INTO social.users JSON '{ "id" : "f6071e72-48ec-4fcb-
bf3e-379c8a696488", "name" : "Jo Quux"}';
```

```
SELECT JSON * FROM social.users;
```

When working with results in JSON format, DevCenter displays them in a new **Results** viewer that support syntax highlighting and code folding (using the +/- signs). You can also copy data as JSON directly from the **Results** panel.

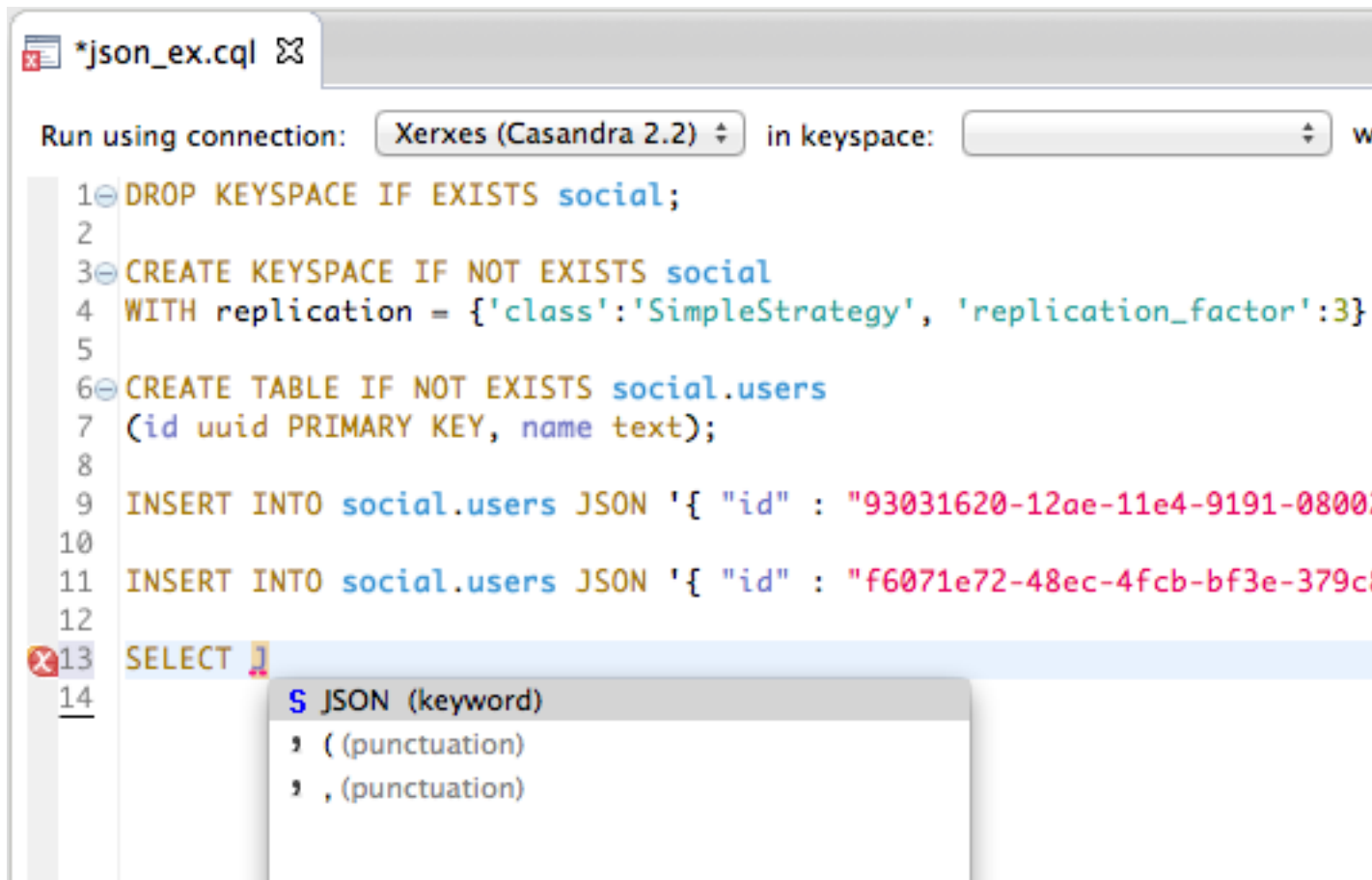


CQL Script editor JSON support

The **CQL Script** editor supports the following when editing JSON-specific statements:

- new CQL keywords and features
- code assist
- syntax highlighting
- validation
- quick fixes

For example, code assist below:



The CQL Script editor also supports the `toJson` and `fromJson` functions.

The `toJson` function is only valid in the selection clause of a `SELECT` statement. For example, given the following schema and some data:

```

CREATE TYPE complex.phone (alias text, number text);

CREATE TYPE complex.address (street text, city text, zip_code int, phones
  list<FROZEN<phone>>);

CREATE TABLE IF NOT EXISTS complex.users
(id uuid PRIMARY KEY, name text, addresses map<text, FROZEN<address>>);

INSERT INTO complex.users (id, name, addresses)
VALUES (756716f7-2e54-4715-9f00-91dcbea6cf50, 'John Doe',
{ 'home': { street : '1021 West 4th St. #202',
            city : 'San Pedro',
            zip_code : 92330,
            phones : [ { alias : 'home', number : '213-555-1212' } ] } } );

INSERT INTO complex.users (id, name, addresses)
VALUES (f6071e72-48ec-4fcb-bf3e-379c8a696488, 'Jane Quux',
{ 'home': { street : '2580 Arnold Dr.',
            city : 'San Fransisco',
            zip_code : 94110,
            phones : [ { alias : 'home', number : '415-555-8945' } ] } } );

INSERT INTO complex.users (id, name, addresses)
VALUES (93031620-12ae-11e4-9191-0800200c9a66, 'Gary Binary',

```



```
{ 'home': { street : '123 Eddy St.',
             city : 'Petaluma',
             zip_code : 95566,
             phones : [ { alias : 'home', number : '707-555-2323' } ] } } );
```

Retrieve the addresses column back as JSON data:

```
SELECT name, toJson(addresses) FROM complex.users;
```

In the **Results** tab:

Results	Query Trace
name	system.toJson(addresses)
Gary Binary	{"home": {"street": "123 Eddy St.", "city": "Petaluma", "zip_code": 95566,
John Doe	{"home": {"street": "1021 West 4th St. #202", "city": "San Pedro", "zip_co
Jane Quux	{"home": {"street": "2580 Arnold Dr.", "city": "San Fransisco", "zip_code":

And, in the **Details** data viewer:

Outline

Details

Type: varchar

Timestamp: [Not available]

TTL: [Not available]

Value:

```
{
  "home" : {
    "street" : "123 Eddy St.",
    "city" : "Petaluma",
    "zip_code" : 95566,
    "phones" : [ {
      "alias" : "home",
      "number" : "707-555-2323"
    } ]
  }
}
```

The `fromJson` function is valid in `INSERT`, `UPDATE`, and `DELETE` statements. Using the same schema as above, an example of an `INSERT` statement:

```
INSERT INTO complex.users (id, name, addresses)
VALUES (d7b9b58a-2edc-11e5-a151-feff819cdc9f, 'Berthold Boxcart',
```

```
fromJson('{"home":{"street":"12 Ghort
St.", "city":"Graton", "zip_code":95444, "phones":
[{"alias":"mobile", "number":"707-555-8452"}]}')');
```

Keyboard shortcuts

Macintosh OS X	Windows or Linux	Description
General		
#+N	Ctrl+N	Generic New wizard.
Editing and running CQL scripts		
#+#+N	Ctrl+Shift+N	New CQL script.
#+#+K	Ctrl+Alt+Shift+K	New keyspace.
#+#+T	Ctrl+Alt+Shift+T	New table.
#+#+I	Ctrl+Alt+Shift+I	New Index.
#+#+U	Ctrl+Alt+Shift+U	New User-defined Type.
#+#+V	Ctrl+Alt+Shift+V	New Materialized View.
#+F3	Ctrl+F3	Open CQL script.
DEL	DEL	Delete CQL script.
#+I	Ctrl+I	Open CQL script properties.
#+F11	Alt+F11	Execute current CQL script. (If statements are selected then only execute those statements.)
#+A	Ctrl+A	Select all CQL scripts.
#+Space	Ctrl+Space	Display auto-completion popup.
Connections panel		
#+#+N	Ctrl+Alt+Shift+N	New connection.
#+F3	Ctrl+F3	Open connection.
#+F4	Ctrl+F4	Close connection
#+#+D	Ctrl+Shift+D	Clone connection.
DEL	DEL	Delete connection.
CQL scripts panel		
#+.	Ctrl+.	Jump to next warning / error.
#+#+.	Ctrl+Shift+.	Jump to previous warning / error.
#+1	Ctrl+1	Display quick fixes popup.
#+#+F	Ctrl+Shift+F	Format code.
#+click	Ctrl+click	Multiple row selection in results pane.

Usage data

We, DataStax, would like your help improving the quality and performance of DevCenter. Starting with version 1.4, DevCenter can automatically collect usage information and send it anonymously and securely to DataStax. The data is sent only with [your explicit consent](#).

Anonymous, confidential, and secure reporting

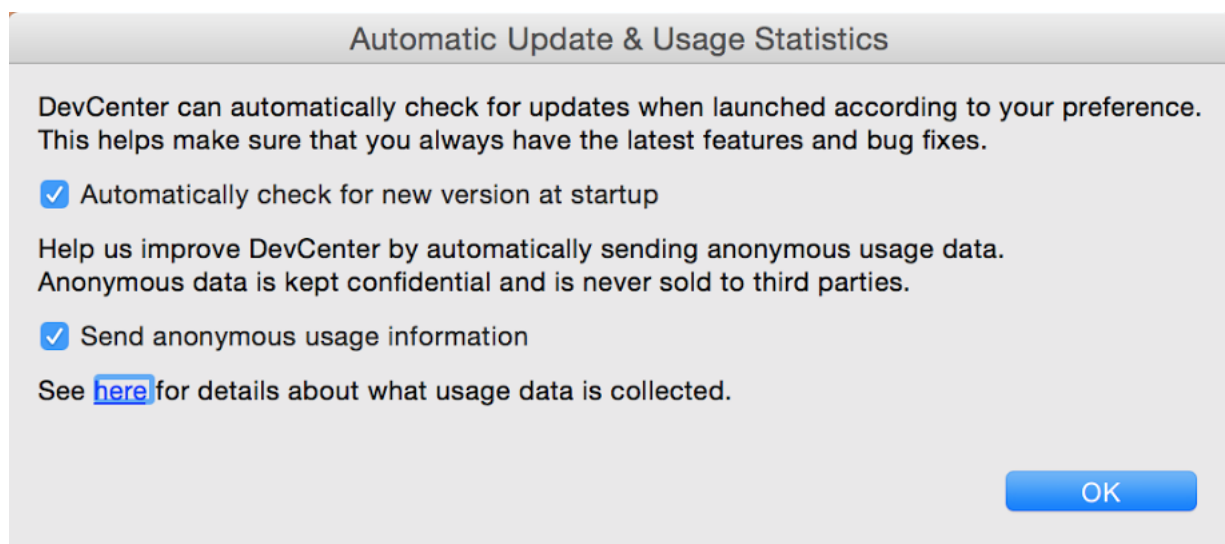
All usage data is collected and sent to DataStax anonymously. None of the information submitted identifies you personally.

Usage data is kept confidential and is never sold to third parties.

Usage data is submitted securely using the HTTPS protocol.

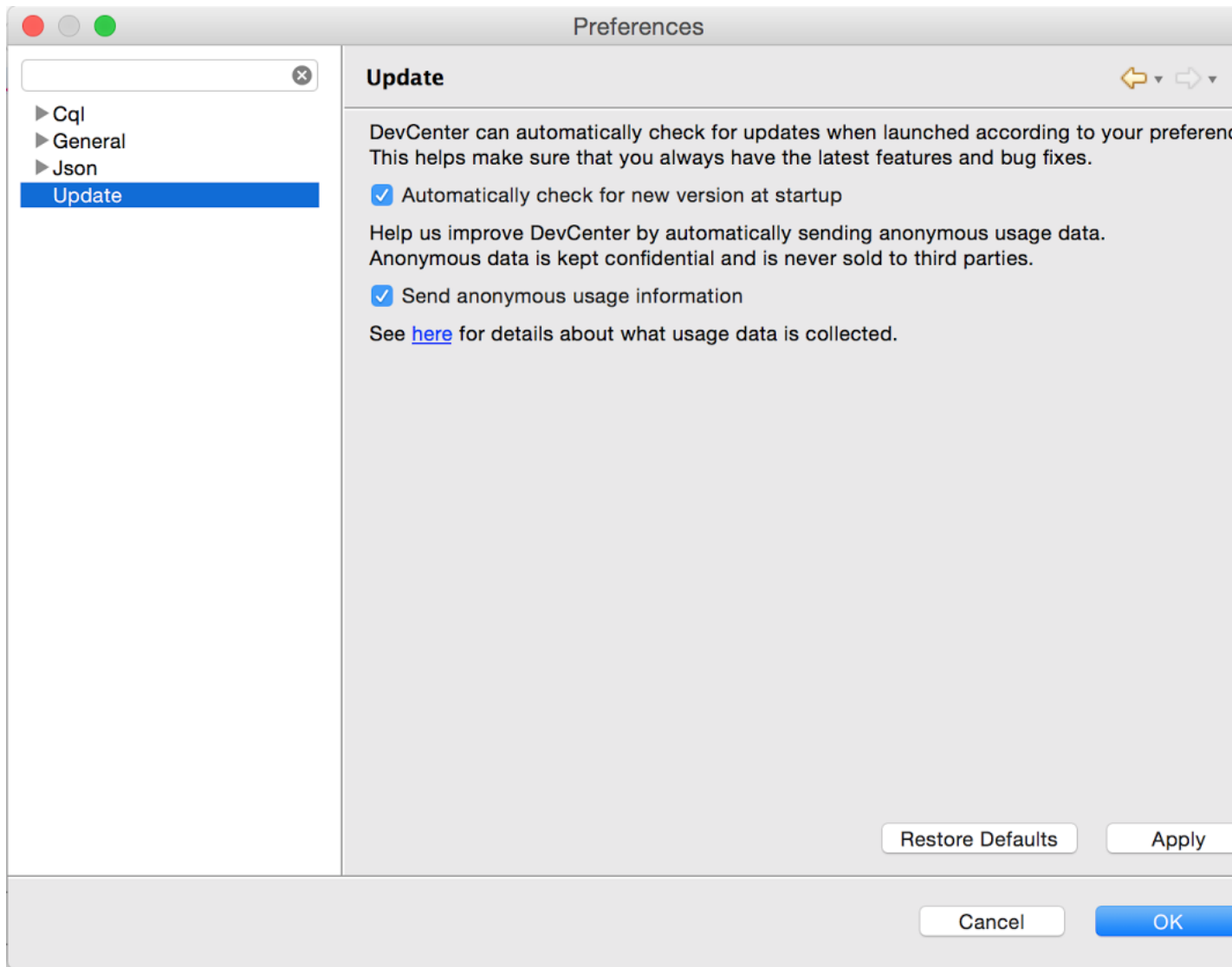
Configuring reporting options

The first time you start DevCenter (after installation or update), you are presented with a first-time **Automatic Update & Usage Statistics** dialog that allows you to opt out of version update and anonymous usage data being collected and sent. Your settings are preserved when upgrading to a new version of DevCenter.



Preferences dialog

The usage data is only sent with your consent and can be turned on and off at any time via the **Preferences** dialog. Select **DevCenter > Preferences > or #-**, (on Mac OS X or **File > Preferences** on Windows or Linux).



The two checkboxes on the **Update** pane allow you to turn on and off:

- version check at startup
- sending anonymous usage data to DataStax

Collected usage data

The following table provides a description of all the information DevCenter is collecting:

Data collected	Description
DevCenter version	The version of the currently running DevCenter. ¹
Java version	The version of Java running DevCenter. ¹
OS version	The version of the operating system running DevCenter. ¹
New Table wizard started	Number of times the New Table wizard has been accessed.
New Table wizard finished	Number of times the New Table wizard has been dismissed.
New Keyspace wizard started	Number of times the New Keyspace wizard has been accessed.

Data collected	Description
New Keyspace wizard finished	Number of times the New Keyspace wizard has been dismissed.
Edit Table wizard started	Number of times the Edit Table wizard has been accessed.
Edit Table wizard finished	Number of times the Edit Table wizard has been dismissed.
Edit Keyspace wizard started	Number of times the Edit Keyspace wizard has been accessed.
Edit Keyspace wizard finished	Number of times the Edit Keyspace wizard has been dismissed.
Clone Table wizard started	Number of times the Clone Table wizard has been accessed.
Clone Table wizard finished	Number of times the Clone Table wizard has been dismissed.
Clone Keyspace wizard started	Number of times the Clone Keyspace wizard has been accessed.
Clone Keyspace wizard finished	Number of times the Clone Keyspace wizard has been dismissed.
Query Trace Tab selected	Number of times the Query Trace tab has been accessed.
Copy cell	Number of times a cell has been copied in the results tab as CSV data.
Copy row as CSV	Number of times a row has been copied in the results tab as CSV data.
Copy all as CSV	Number of times all the data has been copied in the results tab as CSV data.
Copy row as INSERTs	Number of times a row has been copied in the results tab as an INSERT statement.
Copy all as INSERTs	Number of times all the data has been copied in the results tab as INSERT statements.
Open file menu count	The number of times a CQL script file has been opened.
New file menu count	The number of times a new CQL script file has been created.
Delete file menu count	The number of times a CQL Script file has been deleted.
Execute count	The number of times CQL statements have been executed.
Formatting count	The number of times code formatting has been invoked in the CQL Script editor.
Validation error count	The number of times a validation error has been hovered over in the CQL Script editor to see the details.
Quick fix invocation count	The number of times a particular quick fix proposal has been accepted. As with errors there is a counter for each of these. For example if there is an error <code>column_name_does_not_exists</code> , the <code>column_name_does_not_exist_quick_fix</code> counter is incremented if the proposed quick fix for that error has been accepted.
Time used in seconds	The number of seconds that DevCenter has been running.
Maximum connection count	The maximum number of simultaneously open connections since DevCenter was started.
Maximum CQL script count	The maximum number of simultaneously open CQL scripts since DevCenter was started.

Data collected	Description
Maximum table count	The maximum number of tables in any one schema associated with an open connection since DevCenter was started.
Maximum UDT count	The maximum number of user-defined types in any one schema associated with an open connection since DevCenter was started.
Maximum script line count	The maximum number of statements across all CQL Script files open since DevCenter was started
Maximum open editor count	Maximum number of open files observed since DevCenter was started.

¹ Introduced in DevCenter 1.4.1.

User-defined functions and aggregate functions

The **CQL Script** editor supports the creation of user-defined functions (UDF) in CQL and their use in INSERT and SELECT statements.

Assuming a table:

```
CREATE TABLE cycling.cyclist_points (
  id UUID,
  name text,
  race_title text,
  race_points double,
  PRIMARY KEY (id, race_points )
);
```

A UDF can be created and used as follows:

```
CREATE OR REPLACE FUNCTION cycling.flog(d double)
CALLED ON NULL INPUT RETURNS double
LANGUAGE java AS 'return Double.valueOf(Math.log(d.doubleValue()))';

INSERT INTO cycling.cyclist_points
JSON '{
  "id" : "93031620-12ae-11e4-9191-0800200c9a66",
  "name" : "Georgina Bronzini",
  "race_title" : "Tour of Chongming Island World Cup",
  "race_points" : 120
}';

INSERT INTO cycling.cyclist_points
JSON '{
  "id" : "f6071e72-48ec-4fcb-bf3e-379c8a696488",
  "name" : "Paolo Tiralongo",
  "race_title" : "98th Giro d'Italia - Stage 15",
  "race_points" : 2
}';

SELECT id, name, flog(race_points) FROM cycling.cyclist_points;
```

Results	Query Trace	
id	name	cycling.flog(race_p
93031620-12ae-11e4-9191-0800200c9a66	Georgina Bronzini	4.7874917427820
f6071e72-48ec-4fcb-bf3e-379c8a696488	Paolo Tiralongo	0.6931471805599

User-defined aggregate functions

The **CQL Script** editor supports the creation of user-defined aggregate functions (UDAF) in CQL and their use in SELECT statements.

Assuming a table:

```
CREATE TABLE social.numbers (
  id text,
  num int,
  PRIMARY KEY ( id )
);
```

A UDAF can be created and used as follows:

```
INSERT INTO social.numbers (id, num) VALUES ('007', 110);
INSERT INTO social.numbers (id, num) VALUES ('123', 34);
INSERT INTO social.numbers (id, num) VALUES ('111', 667);
INSERT INTO social.numbers (id, num) VALUES ('099', 12);

CREATE OR REPLACE FUNCTION social.my_state ( s int, b int )
CALLED ON NULL INPUT
RETURNS int
LANGUAGE java
AS $$
  if ( s == 0 ) return 1; else return s * b;
$$;

CREATE OR REPLACE AGGREGATE social.my_product ( int )
SFUNC social.my_state
STYPE int
INITCOND 0;

SELECT social.my_product(num) FROM social.numbers;
```

FAQ

Why does DevCenter on Mac OS need Java 6?

When launching DevCenter on Mac OS X, you might see an error message stating: "To open 'DevCenter' you need to install the legacy Java SE 6 runtime. Click 'More Info...' to visit the legacy Java SE 6 download

website". Although DevCenter is compatible with any recent Java version, Mac OS X might force you to use Java SE 6. If this happens, simply follow the link in the message and install Java SE 6.

Why am I seeing the "Failed to load the JNI shared library" error when starting DevCenter?

It's likely to be because you're [using a 32-bit application on a 64-bit JVM](#) or vice versa.

Why aren't all of my database objects showing up in the auto-completion popup?

To retrieve an up-to-date view of the database schema, DevCenter requires an active connection to the cluster, so if you are working offline, only the database objects defined in the current script will be available for auto-completion.

Why isn't the Detailed Results View updated when a new Results table cell is selected?

This is a known issue using VNC Viewer to connect to a virtual machine running DevCenter. Connecting using the VMWare console does not have this problem.

Is there any way to disable Query Tracing?

By default, DevCenter enables query tracing only for the last query executed in a script. A system property can be used to control this by adding the following line to the DevCenter `config.ini` file:

```
devcenter.querytrace.enabled=false
```

The `config.ini` file is located in:

- Mac OS X: `DevCenter_Home/configuration`
- Linux: `DevCenter_Home/configuration`
- Windows: `DevCenter_Home/configuration`

Does DevCenter allow connection to a DSE cluster configured with LDAP authentication?

Yes, DevCenter 1.5.0 has been tested to be able to connect to a DSE cluster configured with OpenLDAP, OracleLDAP, WinAD08 and WinAD12.

I just got "An internal error occurred during: 'Xtext validation', Java heap space" dialog. What can I do about it?

Increase JVM MaxHeapSize by adding `-Xmx512m` or `-Xmx1024m` to the `DevCenter.ini` file:

The `config.ini` file is located in:




- Mac OS X: `DevCenter_Home/configuration`
- Linux: `DevCenter_Home/configuration`
- Windows: `DevCenter_Home/configuration`







Tips for using DataStax documentation

Navigating the documents

To navigate, use the Contents pane or search. Additional controls are:

Toolbar icons

	Go back through the topics as listed in the Contents pane.
	Go forward through the topics as listed in the Contents pane.
	See doc tweets and provide feedback.

	Display PDF version.
	Send email to DataStax docs.
	Print page.
Contents, bookmarking, and legend icons	
	Opens Contents items. Also expands and collapses text, such as Synopsis and Nodetool legends.
	Closes the Contents items.
	Appears on headings for bookmarking. Right-click to get the link.

Searching documents

Search is designed for each product guide. For example, if searching in DataStax Enterprise 4.8, the results include topics from DataStax Enterprise 4.8, Cassandra 2.1, and CQL 3.1. The results are displayed in tabs:

×

All
DataStax Enterprise
Cassandra
CQL

About 6,800 results (0.38 seconds)
Sort by: Relevance ▾

[Initializing a multiple node cluster \(single data center\)](#)
[docs.datastax.com/en/cassandra/2.1/.../initializeSingleDS.html](#)
A deployment scenario for a Cassandra **cluster** with a single data center. | Version 2.1.
Labeled Cassandra

Other resources

You can find more information and help at:

- [Documentation home page](#)
- [DataStax Academy](#)
- [Datasheets](#)
- [Webinars](#)
- [Whitepapers](#)
- [DataStax Developer Blogs](#)
- [Support](#)