### How to Implement a QueryCommand

This section describes how to implement the query command using the [SampleDriverQueryCommand.java](https://github.com/logpresso/logpresso-app-examples/blob/main/logpresso-sample-app/src/main/java/com/logpresso/sonar/sample/query/SampleDriverQueryCommand.java) code and the [SampleSubnetGroupsCommand.java](https://github.com/logpresso/logpresso-app-examples/blob/main/logpresso-sample-app/src/main/java/com/logpresso/sonar/sample/query/SampleSubnetGroupsCommand.java) code.

#### Driver query command

The SampleDriverQueryCommand class inherits from the DriverQueryCommand class and implements the FieldOrdering interface.

public abstract class SampleDriverQueryCommand extends DriverQueryCommand implements FieldOrdering { protected SampleParams params; public SampleDriverQueryCommand(SampleParams params) { this.params = params; } protected abstract void run(ConnectProfile profile) throws IOException; @Override public void run() { for (ConnectProfile profile : params.getProfiles()) { if (isCancelRequested()) return; try { run(profile); } catch (Throwable t) { String msg = String.format("%s (profile %s) error - %s", getName(), profile.getName(), t.getMessage()); throw new IllegalStateException(msg, t); } } }

Among query commands, those that directly launch a thread by implementing run() as above are referred to as driver commands. There cannot be more than one driver command in a query: the driver command is placed at the front, and other commands leading to the pipe are executed as onPush() is called by the driver command's thread. Additional driver commands can be used in subqueries, such as join or union.

Query commands must be careful not to miss implementing cancel. Loops should continuously check to see if a cancel has been requested by the current user or system by calling isCancelRequested(), and if so, abort immediately.

As a parameter to the constructor, we are passed a SampleParams object, which is a class that inherits from ConnectProfileParams, so we can get a list of connect profile objects.

public String toString() { String s = getName(); if (params.getName() != null) s += Strings.doubleQuote(params.getName()); return s;}

All query commands (QueryCommand) must implement toString(). The string representation of a QueryCommand must follows the valid query command syntax in a normalized form that can be re-parsed by the query parser.

This is because there is a step in generating the distributed query execution plan that converts the QueryCommand object back to a query string literal. If this implementation is missing, the execution plan will not display correctly, and the distributed query may fail.

#### Query command details

The SampleSubnetGroupsCommand inherits from the SampleDriverQueryCommand and implements the following methods:

* getName(): Name of the query command.
* getFieldOrder(): Field display order. If a field does not exist in the query results, it is ignored, which means that here you define a complete field order for the set of all possible field names that could be output. On the screen, the field output order of the last command that implements the FieldOrdering interface in the order of the query commands is applied.
* run(): Implements a function to be executed by the thread of the driver command.

The example code uses the RestApiClient class to call the REST API and calls the pushPipe() method to output the result of the command. The parameters of pushPipe() use the Row class, which is an implementation of the Map interface, so you can pass variable key-value pairs. However, only the following types are allowed for values:

|  |  |  |
| --- | --- | --- |
| Java type | Logpresso type | Description |
| java.lang.String | string | String |
| java.lang.Short | short | 16bit integer number |
| java.lang.Integer | int | 32bit integer number |
| java.lang.Long | long | 64bit integer |
| java.lang.Float | float | 32bit real number |
| java.lang.Double | double | 64bit real number |
| java.lang.Boolean | bool | Boolean |
| java.util.Date | date | Date and time |
| java.net.Inet4Address | ipv4 | IPv4 address |
| java.net.Inet6Address | ipv6 | IPv6 address |
| java.util.List | list | List |
| java.util.Map | map | Map. Only string type is allowed for key. |
| byte | binary | Byte array |

If you need to process more than a few hundred thousand records per second, we recommend calling the pushPipe(VectorizedRowBatch) method to output the data. The following vector types can be used for the value of VectorizedRowBatch.

|  |  |  |
| --- | --- | --- |
| Vector type | Logpresso type | Description |
| org.araqne.log.api.ObjectVector | object | String or any types. |
| org.araqne.log.api.IntVector | int | Optimized for 32bit integer number array. |
| org.araqne.log.api.LongVector | long | Optimized for 64bit integer number array. |
| org.araqne.log.api.DoubleVector | double | Optimized for 64bit real number array. |
| org.araqne.log.api.BooleanVector | bool | Optimized for boolean array. |

Vectorizing and batching 1000 records reduces the overhead of calling the `pushPipe()' function by a factor of 1000, minimizes performance degradation due to lock contention, and reduces garbage collection, resulting in high performance.