### groovy

Runs a script written in Groovy.

#### Syntax

groovy CLASS\_NAME

Required Parameter

**CLASS\_NAME**

Name of the class you want to execute.

#### Description

Groovy is a dynamic object-oriented language developed with influence from languages such as Python and Ruby and runs on the JVM. The script file must meet the following constraints to be executed:

* You need to specify the script file name in the following format: CLASS\_NAME.groovy
* The script files SHOULD be in **data/araqne-logdb-groovy/query\_scripts** in the directory where Logpresso is installed.
* You need to import and use the package provided by Logpresso. Use the following packages as needed.
* groovy.transform.CompileStatic
* org.araqne.logdb.groovy.GroovyQueryScript (required)
* org.araqne.logdb.QueryStopReason
* org.araqne.logdb.Row (required)
* org.araqne.logdb.RowBatch
* org.araqne.logdb.RowPipe

To improve the performance of Groovy scripts, refer to the following:

* Avoid using string processing methods as much as possible. As the number of string objects increases, garbage collection occurs more frequently in the JVM.
* Avoid using the split() and tokenize() methods as much as possible.
* split() is very slow because it uses a regular expression internally.
* Use indexOf() or substring() instead. Although the code is longer, it provides better processing performance.
* Avoid using Pattern.compile(). Reusing the Matcher instance by calling Matcher.reset() provides better performance.
* DO NOT use exceptions for the normal flow of control
* If exceptions occur frequently, processing performance becomes significantly slower.
* If possible, handle possible error cases through conditional testing.

#### Usage

Save the following script as **ToAscii.groovy** in **data/araqne-logdb-groovy/query\_scripts** in the directory where Logpresso is installed.

import groovy.transform.CompileStatic;import org.araqne.logdb.Row;import org.araqne.logdb.groovy.GroovyQueryScript;@CompileStatic(groovy.transform.TypeCheckingMode.SKIP)class ToAscii extends GroovyQueryScript { def void onRow(Row row) { byte[] payload = row.get('payload') char[] chars = new char[payload.length]; for (int i = 0; i < payload.length; i++) { char c = (char) payload[i] if (c < 32 || c > 126) c = '.' chars[i] = c } row.put('text', new String(chars)) pipe.onRow(row) }}

This script encodes the 32nd to 127th characters in ASCII format among the binary values decoded in the PCAP file and assigned on the **payload** field.

pcapfile /opt/logpresso/sonar/http-2.pcap | pcapdecode | groovy ToAscii