generate log-summary

In this article

[Advanced] Create a summary of a structured log file.

GitHub CodeQL is licensed on a per-user basis upon installation. You can use CodeQL only for certain tasks under the license restrictions. For more information, see "About the CodeQL CLI." If you have a GitHub Advanced Security license, you can use CodeQL for automated analysis, continuous integration, and continuous delivery. For more information, see "About GitHub Advanced Security."

This content describes the most recent release of the CodeQL CLI. For more information about this release, see https://github.com/github/codeql-cli-binaries/releases.

To see details of the options available for this command in an earlier release, run the command with the `--help` option in your terminal.

Synopsis

Shell

```
codeql generate log-summary <options>... -- <input> <result>
```

Description

[Advanced] Create a summary of a structured log file.

This command creates a summary of a structured JSON evaluator event log. The output of this command aims to be more stable across different versions of the CLI than the log files themselves. Thus, when implementing a script that uses output from the logs, it is strongly recommended to run this command and use its output rather than using the event logs directly.

Options

Primary Options

`<input>`

[Mandatory] Path to the event log file to produce a summary of.

`<result>`

Path to the location to output the summarised log file to. If this omitted, then the summary will be output to stdout.

`--minify-output`

Where applicable, omit whitespace in the outputted summary. The result will be less human-readable but take up less memory. This option only has an effect for some output formats.

`--utc`

[Advanced] Certain timestamps in the summaries produced by this command may use the local timezone of the machine they are running on. Enabling this flag forces all timestamps to be UTC.

`--format=<format>`

Control the format of the output produced.

`predicates` *(default)*: Produce a summary of the computation performed for each predicate. This will be a stream of JSON objects separated either by two newline characters (by default) or one if the `--minify-output` option is passed.

`text`: Produce a human-readable summary of the evaluation run.

`overall`: Produce a JSON file containing some overall information about the evaluation run, including some summary statistics and information about the most time-consuming evaluations that were performed.

`--[no-]deduplicate-stage-summaries`

[Advanced] This option only works in conjunction with the text format. If passed, this will result in the summary tables containing the most expensive predicates not being repeated for stages that are shared between queries. This has the side-effect of moving all the summary tables to the end of the log, rather than having the ones for each query appear at the point when that query finished.

Common options

`-h, --help`

Show this help text.

`-J=<opt>`

[Advanced] Give option to the JVM running the command.

(Beware that options containing spaces will not be handled correctly.)

`-v, --verbose`

Incrementally increase the number of progress messages printed.

`-q, --quiet`

Incrementally decrease the number of progress messages printed.

`--verbosity=<level>`

[Advanced] Explicitly set the verbosity level to one of errors, warnings, progress, progress+, progress++, progress+++. Overrides `-v` and `-q`.

`--logdir=<dir>`

[Advanced] Write detailed logs to one or more files in the given directory, with generated names that include timestamps and the name of the running subcommand.

(To write a log file with a name you have full control over, instead give `--log-to-stderr` and redirect stderr as desired.)