bqrs interpret

In this article

[Plumbing] Interpret data in a single BQRS.

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 bqrs interpret --format=<format> --output=<output> -t=<String=String> [--threads=<num>] [--source-archive=<sourceArchive>] [--source-location-prefix=<sourceLocationPrefix>] <options>... -- <bqrs-file>
```

Description

[Plumbing] Interpret data in a single BQRS.

A command that interprets a single BQRS file according to the provided metadata and generates output in the specified format.

Options

Primary Options

`<bqrs-file>`

[Mandatory] The BQRS file to interpret.

`--format=<format>`

[Mandatory] The format in which to write the results. One of:

`csv`: Formatted comma-separated values, including columns with both rule and alert metadata.

`sarif-latest`: Static Analysis Results Interchange Format (SARIF), a JSON-based format for describing static analysis results. This format option uses the most recent supported version (v2.1.0). This option is not suitable for use in automation as it will produce different versions of SARIF between different CodeQL versions.

`sarifv2.1.0`: SARIF v2.1.0.

`graphtext`: A textual format representing a graph. Only compatible with queries with @kind graph.

`dgml`: Directed Graph Markup Language, an XML-based format for describing graphs. Only compatible with queries with @kind graph.

`dot`: Graphviz DOT language, a text-based format for describing graphs. Only compatible with queries with @kind graph.

`-o, --output=<output>`

[Mandatory] The output path to write results to. For graph formats this should be a directory, and the result (or results if this command supports interpreting more than one query) will be written within that directory.

`-t=<String=String>`

[Mandatory] A query metadata key value pair. Repeat for each piece of metadata. At least the keys 'kind' and 'id' must be specified. Keys do not need to be prefixed with @.

`--max-paths=<maxPaths>`

The maximum number of paths to produce for each alert with paths. (Default: 4)

`--[no-]sarif-add-file-contents`

[SARIF formats only] Include the full file contents for all files referenced in at least one result.

`--[no-]sarif-add-snippets`

[SARIF formats only] Include code snippets for each location mentioned in the results, with two lines of context before and after the reported location.

`--[no-]sarif-add-query-help`

[SARIF formats only] Include Markdown query help in the results. It loads query help for /path/to/query.ql from the /path/to/query.md file. This option has no effect when passed to codeql bqrs interpret.

`--[no-]sarif-group-rules-by-pack`

[SARIF formats only] Place the rule object for each query under its corresponding QL pack in the `<run>.tool.extensions` property. This option has no effect when passed to codeql bqrs interpret.

`--[no-]sarif-multicause-markdown`

[SARIF formats only] For alerts that have multiple causes, include them as a Markdown-formatted itemized list in the output in addition to as a plain string.

`--no-group-results`

[SARIF formats only] Produce one result per message, rather than one result per unique location.

`--csv-location-format=<csvLocationFormat>`

The format in which to produce locations in CSV output. One of: uri, line-column, offset-length. (Default: line-column)

`--dot-location-url-format=<dotLocationUrlFormat>`

A format string defining the format in which to produce file location URLs in DOT output. The following place holders can be used {path} {start:line} {start:column} {end:line} {end:column}, {offset}, {length}

`--sarif-category=<category>`

[SARIF formats only] Specify a category for this analysis to include in the SARIF output. A category can be used to distinguish multiple analyses performed on the same commit and repository, but on different languages or different parts of the code.

If you analyze the same version of a code base in several different ways (e.g., for different languages) and upload the results to GitHub for presentation in Code Scanning, this value should differ between each of the analyses, which tells Code Scanning that the analyses *supplement* rather than *supersede* each other. (The values should be consistent between runs of the same analysis for *different* versions of the code base.)

This value will appear (with a trailing slash appended if not already present) as the `<run>.automationId` property in SARIF v1, the `<run>.automationLogicalId` property in SARIF v2, and the `<run>.automationDetails.id` property in SARIF v2.1.0.

`-j, --threads=<num>`

The number of threads used for computing paths.

Defaults to 1. You can pass 0 to use one thread per core on the machine, or -*N* to leave *N* cores unused (except still use at least one thread).

`--sarif-run-property=<String=String>`

[SARIF only] A key value pair to add to the generated SARIF 'run' property bag. Can be repeated.

`--column-kind=<columnKind>`

[SARIF only] The column kind used to interpret location columns. One of: utf8, utf16, utf32, bytes.

`--[no-]unicode-new-lines`

[SARIF only] Whether the unicode newline characters LS (Line Separator, U+2028) and PS (Paragraph Separator, U+2029) are considered as new lines when interpreting

location line numbers.

Source archive options - must be given together or not at all

`-s, --source-archive=<sourceArchive>`

The directory or zip file containing the source archive.

`-p, --source-location-prefix=<sourceLocationPrefix>`

The file path on the original file system where the source code was stored.

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.)