

database interpret-results

In this article

- Synopsis
- Description
- Options

[Plumbing] Interpret computed query results into meaningful formats such as SARIF or CSV.

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 database interpret-results --format=<format> --output=<output> [--threads=<num>] <options>... -- <database> <file|dir|suite>...
```

Description

[Plumbing] Interpret computed query results into meaningful formats such as SARIF or CSV.


The results should have been computed and stored in a CodeQL database directory using [codeql database run-queries](#). (Usually you'd want to do these steps together, by using [codeql database analyze](#)).

Options

Primary Options

`<database>` 

[Mandatory] Path to the CodeQL database that has been queried.

<filesuite>... 

Repeat the specification of which queries were executed here.

If omitted, the CLI will determine a suitable set of queries using the same logic as [codeql database run-queries](#).

(In a future version it ought to be possible to omit this and instead interpret all results that are found in the database. That glorious future is not yet. Sorry.)

--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.

--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).

--[no-]print-diagnostics-summary 

Print a summary of the analyzed diagnostics to standard output.

--[no-]print-metrics-summary 

Print a summary of the analyzed metrics to standard output.

`--[no-]print-baseline-loc` [↗](#)

Print the baseline lines of code counted to standard output.

Options for configuring the CodeQL package manager [↗](#)

`--registries-auth-stdin` [↗](#)

Authenticate to GitHub Enterprise Server Container registries by passing a comma-separated list of <registry_url>=<token> pairs.

For example, you can pass

```
https://containers.GHEHOSTNAME1/v2/=TOKEN1,https://containers.GHEHOSTNAME2/v2/=TOKEN2
```

 to authenticate to two GitHub Enterprise Server instances.

This overrides the CODEQL_REGISTRIES_AUTH and GITHUB_TOKEN environment variables. If you only need to authenticate to the github.com Container registry, you can instead authenticate using the simpler `--github-auth-stdin` option.

`--github-auth-stdin` [↗](#)

Authenticate to the github.com Container registry by passing a github.com GitHub Apps token or personal access token via standard input.

To authenticate to GitHub Enterprise Server Container registries, pass `--registries-auth-stdin` or use the CODEQL_REGISTRIES_AUTH environment variable.

This overrides the GITHUB_TOKEN environment variable.

Options for finding QL packs (which may be necessary to interpret query suites) [↗](#)

`--search-path=<dir>[:<dir>...]` [↗](#)

A list of directories under which QL packs may be found. Each directory can either be a QL pack (or bundle of packs containing a `.codeqlmanifest.json` file at the root) or the immediate parent of one or more such directories.

If the path contains more than one directory, their order defines precedence between them: when a pack name that must be resolved is matched in more than one of the directory trees, the one given first wins.

Pointing this at a checkout of the open-source CodeQL repository ought to work when querying one of the languages that live there.

If you have checked out the CodeQL repository as a sibling of the unpacked CodeQL toolchain, you don't need to give this option; such sibling directories will always be searched for QL packs that cannot be found otherwise. (If this default does not work, it is strongly recommended to set up `--search-path` once and for all in a per-user configuration file).

(Note: On Windows the path separator is `;`).

`--additional-packs=<dir>[:<dir>...]` [↗](#)

If this list of directories is given, they will be searched for packs before the ones in `--search-path`. The order between these doesn't matter; it is an error if a pack name is found in two different places through this list.

This is useful if you're temporarily developing a new version of a pack that also appears

in the default path. On the other hand, it is *not recommended* to override this option in a config file; some internal actions will add this option on the fly, overriding any configured value.

(Note: On Windows the path separator is `;`).

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.)

Legal

© 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Status](#) [Pricing](#) [Expert services](#) [Blog](#)