test accept

In this article

Accept results of failing unit tests.

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 test accept <options>... -- <test|dir>...
```

Description

Accept results of failing unit tests.

This is a convenience command that renames the `.actual` files left by codeql test run for failing tests into `.expected`, such that future runs on the tests that give the same output will be considered to pass. What it does can also be achieved by ordinary file manipulation, but you may find its syntax more useful for this special case.

The command-line arguments specify one or more *tests* -- that is, `.ql(ref)` files -- and the command automatically derives the names of the `.actual` files from them. Any test that doesn't have an `.actual` file will be silently ignored, which makes it easy to accept just the results of *failing* tests from a previous run.

Options

Primary Options

`<test|dir>...`

Each argument is one of:

- A `.ql` or `.qlref` file that defines a test to run.
- A directory which will be searched recursively for tests to run.

`--slice=<N/M>`

[Advanced] Divide the test cases into *M* roughly equal-sized slices and process only the _N_th of them. This can be used for manual parallelization of the testing process.

`--[no-]strict-test-discovery`

[Advanced] Only use queries that can be strongly identified as tests. This mode tries to distinguish between `.ql` files that define unit tests and `.ql` files that are meant to be useful queries. This option is used by tools, such as IDEs, that need to identify all unit tests in a directory tree without depending on previous knowledge of how the files in it are arranged.

Within a QL pack whose `qlpack.yml` declares a `tests` directory, all `.ql` files in that directory are considered tests, and `.ql` files outside it are ignored. In a QL pack that doesn't declare a `tests` directory, a `.ql` file is identified as a test only if it has a corresponding `.expected` file.

For consistency, `.qlref` files are limited by the same rules as `.ql` files even though a `.qlref` file cannot really be a non-test.

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.)