resolve upgrades

In this article

[Deep plumbing] Determine upgrades to run for a raw dataset.

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 resolve upgrades --dbscheme=<file> <options>...
```

Description 🔗

[Deep plumbing] Determine upgrades to run for a raw dataset.

Determine which upgrades need to be performed on a particular raw QL dataset to bring it up to the state of the configured QL libraries. This computation is part of what happens during an ordinary database upgrade, and is exposed as a separate plumbing command in order to (a) help with troubleshooting, and (b) provide a starting point for modifying the path in extraordinary cases where exact control is needed.

Options 🔗

Primary Options 🔗

`--dbscheme=<file>` 🔗

[Mandatory] The *current* dbscheme of the dataset we want to upgrade.

`--format=<fmt>` 🔗

Select output format. Choices include:

`lines` *(default)*: Print upgrade scripts on one line each.

`json` : Print a JSON array of upgrade script paths.

`--just-check` 🔗

Don't print any output, but exit with code 0 if there are upgrades to do, and code 1 if there are none.

Options from the invoking command's command line 🔗

`--search-path=<dir>[:<dir>...]` 🔗

A list of directories under which QL packs containing upgrade recipes 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 directories trees, their order defines precedence between them: if 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.

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

`--additional-packs=<dir>[:<dir>...]` 🔗

[Advanced] If this list of directories is given, they will be searched for upgrades 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 `;`).

`--target-dbscheme=<file>` 🔗

The *target* dbscheme we want to upgrade to. If this is not given, a maximal upgrade path will be constructed

`--target-sha=<sha>` 🔗

[Advanced] An alternative to `--target-dbscheme` that gives the internal hash of the target dbscheme instead of the dbscheme file.

`--[no-]allow-downgrades` 🔗

Include any relevant downgrades if there are no upgrades

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.)