



Linux Ubuntu 26.04 Manual Pages on command 'npm-exec.1'

\$ man npm-exec.1

NPM-EXEC(1)

General Commands Manual

NPM-EXEC(1)

NAME

npm-exec

Synopsis

<!-- AUTOGENERATED USAGE DESCRIPTIONS -->

Description

This command allows you to run an arbitrary command from an npm package (either one installed locally, or fetched remotely), in a similar context as running it via npm run.

Run without positional arguments or --call, this allows you to interactively run commands in the same sort of shell environment that package.json scripts are run. Interactive mode is not supported in CI environments when standard input is a TTY, to prevent hangs.

Whatever packages are specified by the --package option will be provided in the PATH of the executed command, along with any locally installed package executables. The --package option may be specified multiple times, to execute the supplied command in an environment where all specified packages are available.

If any requested packages are not present in the local project dependencies, then a prompt is printed, which can be suppressed by providing either --yes or --no. When standard input is not a TTY or a CI environment is detected, --yes is assumed. The requested packages are installed to a folder in the npm cache, which is added to the PATH

environment variable in the executed process.

Package names provided without a specifier will be matched with whatever version exists in the local project. Package names with a specifier will only be considered a match if they have the exact same name and version as the local dependency.

If no `-c` or `--call` option is provided, then the positional arguments are used to generate the command string. If no `--package` options are provided, then npm will attempt to determine the executable name from the package specifier provided as the first positional argument according to the following heuristic:

- ? If the package has a single entry in its `bin` field in `package.json`, or if all entries are aliases of the same command, then that command will be used.

- ? If the package has multiple `bin` entries, and one of them matches the unscoped portion of the name field, then that command will be used.

- ? If this does not result in exactly one option (either because there are no `bin` entries, or none of them match the name of the package), then `npm exec` exits with an error.

To run a binary other than the named binary, specify one or more `--package` options, which will prevent npm from inferring the package from the first command argument.

`npx` vs `npm exec`

When run via the `npx` binary, all flags and options must be set prior to any positional arguments. When run via `npm exec`, a double-hyphen `--` flag can be used to suppress npm's parsing of switches and options that should be sent to the executed command.

For example:

```
$ npx foo@latest bar --package=@npmcli/foo
```

In this case, npm will resolve the `foo` package name, and run the following command:

```
$ foo bar --package=@npmcli/foo
```

Since the `--package` option comes after the positional arguments, it is treated as an argument to the executed command.

In contrast, due to npm's argument parsing logic, running this command is different:

```
$ npm exec foo@latest bar --package=@npmcli/foo
```

In this case, npm will parse the `--package` option first, resolving the `@npmcli/foo` package. Then, it will execute the following command in that context:

```
$ foo@latest bar
```

The double-hyphen character is recommended to explicitly tell npm to stop parsing command line options and switches. The following command would thus be equivalent to the `npx` command above:

```
$ npm exec -- foo@latest bar --package=@npmcli/foo
```

Configuration

```
<!-- AUTOGENERATED CONFIG DESCRIPTIONS -->
```

Examples

Run the version of `tap` in the local dependencies, with the provided arguments:

```
$ npm exec -- tap --bail test/foo.js
```

```
$ npx tap --bail test/foo.js
```

Run a command other than the command whose name matches the package name by specifying a `--package` option:

```
$ npm exec --package=foo -- bar --bar-argument
```

```
# ~ or ~
```

```
$ npx --package=foo bar --bar-argument
```

Run an arbitrary shell script, in the context of the current project:

```
$ npm x -c 'eslint && say "hooray, lint passed"'
```

```
$ npx -c 'eslint && say "hooray, lint passed"'
```

Workspaces support

You may use the workspace or workspaces configs in order to run an arbitrary command from an npm package (either one installed locally, or fetched remotely) in the context of the specified workspaces.

If no positional argument or `--call` option is provided, it will open an interactive subshell in the context of each of these configured workspaces one

at a time.

Given a project with configured workspaces, e.g:

```
.
+-- package.json
`-- packages
    +-- a
    |   `-- package.json
    +-- b
    |   `-- package.json
    `-- c
        `-- package.json
```

Assuming the workspace configuration is properly set up at the root level

package.json file. e.g:

```
{
  "workspaces": [ "./packages/*" ]
}
```

You can execute an arbitrary command from a package in the context of each of the configured workspaces when using the workspaces config options, in this example

we're using eslint to lint any js file found within each workspace folder:

```
npm exec --ws -- eslint ./*.js
```

Filtering workspaces

It's also possible to execute a command in a single workspace using the workspace config along with a name or directory path:

```
npm exec --workspace=a -- eslint ./*.js
```

The workspace config can also be specified multiple times in order to run a specific script in the context of multiple workspaces. When defining values for the workspace config in the command line, it's also possible to use -w as a shorthand, e.g:

```
npm exec -w a -w b -- eslint ./*.js
```

This last command will run the eslint command in both ./packages/a and ./packages/b folders.

The npx binary was rewritten in npm v7.0.0, and the standalone npx package deprecated at that time. npx uses the npm exec command instead of a separate argument parser and install process, with some affordances to maintain backwards compatibility with the arguments it accepted in previous versions.

This resulted in some shifts in its functionality:

- ? Any npm config value may be provided.
- ? To prevent security and user-experience problems from mistyping package names, npx prompts before installing anything. Suppress this prompt with the -y or --yes option.
- ? The --no-install option is deprecated, and will be converted to --no.
- ? Shell fallback functionality is removed, as it is not advisable.
- ? The -p argument is a shorthand for --parseable in npm, but shorthand for --package in npx. This is maintained, but only for the npx executable.
- ? The --ignore-existing option is removed. Locally installed bins are always present in the executed process PATH.
- ? The --npm option is removed. npx will always use the npm it ships with.
- ? The --node-arg and -n options are removed.
- ? The --always-spawn option is redundant, and thus removed.
- ? The --shell option is replaced with --script-shell, but maintained in the npx executable for backwards compatibility.

A note on caching

The npm cli utilizes its internal package cache when using the package name specified. You can use the following to change how and when the cli uses this cache. See npm cache for more on how the cache works.

prefer-online

Forces staleness checks for packages, making the cli look for updates immediately even if the package is already in the cache.

prefer-offline

Bypasses staleness checks for packages. Missing data will still be

requested from the server. To force full offline mode, use offline.

offline

Forces full offline mode. Any packages not locally cached will result in an error.

workspace

? Default:

? Type: String (can be set multiple times)

Enable running a command in the context of the configured workspaces of the current project while filtering by running only the workspaces defined by this configuration option.

Valid values for the workspace config are either:

? Workspace names

? Path to a workspace directory

? Path to a parent workspace directory (will result to selecting all of the nested workspaces)

This value is not exported to the environment for child processes.

workspaces

? Alias: --ws

? Type: Boolean

? Default: false

Run scripts in the context of all configured workspaces for the current project.

See Also

? npm run-script

? npm scripts

? npm test

? npm start

? npm restart

? npm stop

? npm config

? npm workspaces

? npx