



**Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!**

***Rocky Enterprise Linux 9.2 Manual Pages on command 'npm-dist-tag.1'***

***\$ man npm-dist-tag.1***

NPM-DIST-TAG(1)

NPM-DIST-TAG(1)

NAME

npm-dist-tag - Modify package distribution tags

Synopsis

npm dist-tag add <pkg>@<version> [<tag>]

npm dist-tag rm <pkg> <tag>

npm dist-tag ls [<pkg>]

aliases: dist-tags

Description

Add, remove, and enumerate distribution tags on a package:

? add: Tags the specified version of the package with the specified tag, or the --tag con?

fig if not specified. If you have two-factor authentication on auth-and-writes then

you?ll need to include a one-time password on the command line with --otp <one-time password>, or at the OTP prompt.

? rm: Clear a tag that is no longer in use from the package. If you have two-factor au?

thentication on auth-and-writes then you?ll need to include a one-time password on the command line with --otp <one-time password>, or at the OTP prompt.

? ls: Show all of the dist-tags for a package, defaulting to the package in the current prefix. This is the default action if none is specified.

A tag can be used when installing packages as a reference to a version instead of using a specific version number:

npm install <name>@<tag>

When installing dependencies, a preferred tagged version may be specified:

```
npm install --tag <tag>
```

(This also applies to any other commands that resolve and install dependencies, such as `npm dedupe`, `npm update`, and `npm audit fix`.)

Publishing a package sets the latest tag to the published version unless the `--tag` option is used. For example, `npm publish --tag=beta`.

By default, `npm install <pkg>` (without any `@<version>` or `@<tag>` specifier) installs the latest tag.

## Purpose

Tags can be used to provide an alias instead of version numbers.

For example, a project might choose to have multiple streams of development and use a different tag for each stream, e.g., `stable`, `beta`, `dev`, `canary`.

By default, the latest tag is used by npm to identify the current version of a package, and `npm install <pkg>` (without any `@<version>` or `@<tag>` specifier) installs the latest tag. Typically, projects only use the latest tag for stable release versions, and use other tags for unstable versions such as prereleases.

The next tag is used by some projects to identify the upcoming version.

Other than latest, no tag has any special significance to npm itself.

## Caveats

This command used to be known as `npm tag`, which only created new tags, and so had a different syntax.

Tags must share a namespace with version numbers, because they are specified in the same slot: `npm install <pkg>@<version>` vs `npm install <pkg>@<tag>`.

Tags that can be interpreted as valid semver ranges will be rejected. For example, `v1.4` cannot be used as a tag, because it is interpreted by semver as `>=1.4.0 <1.5.0`. See <https://github.com/npm/npm/issues/6082>.

The simplest way to avoid semver problems with tags is to use tags that do not begin with a number or the letter `v`.

## Configuration

```
<!-- AUTOGENERATED CONFIG DESCRIPTIONS START --> <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->
```

## 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 in selecting all workspaces within that folder)

When set for the npm init command, this may be set to the folder of a workspace which does not yet exist, to create the folder and set it up as a brand new workspace within the project.

This value is not exported to the environment for child processes. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

#### workspaces

? Default: null

? Type: null or Boolean

Set to true to run the command in the context of all configured workspaces.

Explicitly setting this to false will cause commands like install to ignore workspaces altogether. When not set explicitly:

? Commands that operate on the node\_modules tree (install, update, etc.) will link workspaces into the node\_modules folder. - Commands that do other things (test, exec, publish, etc.) will operate on the root project, unless one or more workspaces are specified in the workspace config.

This value is not exported to the environment for child processes. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

#### include-workspace-root

? Default: false

? Type: Boolean

Include the workspace root when workspaces are enabled for a command.

When false, specifying individual workspaces via the workspace config, or all workspaces via the workspaces flag, will cause npm to operate only on the specified workspaces, and not on the root project. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

<!-- AUTOGENERATED CONFIG DESCRIPTIONS END -->

#### See Also

? npm help publish

? npm help install

? npm help dedupe

? npm help registry

? npm help config

? npm help npmrc

undefined NaN

NPM-DIST-TAG(1)