Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'npm-pkg.1' command

$ man npm-pkg.1

NPM-PKG(1) NPM-PKG(1)

NAME

npm-pkg - Manages your package.json

Synopsis

npm pkg set <key>=<value> [<key>=<value> ...]

npm pkg get [<key> [<key> ...]]

npm pkg delete <key> [<key> ...]

npm pkg set [<array>[<index>].<key>=<value> ...]

npm pkg set [<array>[].<key>=<value> ...]

Description

A command that automates the management of package.json files. npm pkg provide 3 different sub commands that allow you to modify or retrieve values for given object keys in your package.json.

The syntax to retrieve and set fields is a dot separated representation of the nested object properties to be found within your package.json, it's the same notation used in npm help view to retrieve information from the registry manifest, below you can find more examples on how to use it.

Returned values are always in json format.

? npm pkg get <field>

Retrieves a value key, defined in your package.json file.

For example, in order to retrieve the name of the current package, you

can run:

npm pkg get name

It's also possible to retrieve multiple values at once:

npm pkg get name version

You can view child fields by separating them with a period. To re?
trieve
the value of a test script value, you would run the following com?
mand:

npm pkg get scripts.test

For fields that are arrays, requesting a non-numeric field will re?
turn
all of the values from the objects in the list. For example, to get
all
the contributor emails for a package, you would run:

npm pkg get contributors.email

You may also use numeric indices in square braces to specifically se?
lect
an item in an array field. To just get the email address of the first
contributor in the list, you can run:

npm pkg get contributors[0].email

For complex fields you can also name a property in square brackets
to specifically select a child field. This is especially helpful
with the exports object:

npm pkg get "exports[.].require"

? npm pkg set <field>=<value>

Sets a value in your package.json based on the field value. When
saving to your package.json file the same set of rules used during
npm install and other cli commands that touches the package.json
file
are used, making sure to respect the existing indentation and pos?
sibly
applying some validation prior to saving values to the file.

The same syntax used to retrieve values from your package can also

be used
to define new properties or overriding existing ones, below are some
examples of how the dot separated syntax can be used to edit your
package.json file.
Defining a new bin named mynewcommand in your package.json that points
to a file cli.js:
npm pkg set bin.mynewcommand=cli.js
Setting multiple fields at once is also possible:
npm pkg set description='Awesome package' engines.node='>=10'
It's also possible to add to array values, for example to add a new
contributor entry:
npm pkg set contributors[0].name='Foo' contributors[0].email='foo@bar.ca'
You may also append items to the end of an array using the special
empty bracket notation:
npm pkg set contributors[].name='Foo' contributors[].name='Bar'
It's also possible to parse values as json prior to saving them to your
package.json file, for example in order to set a "private": true
property:
npm pkg set private=true --json
It also enables saving values as numbers:
npm pkg set tap.timeout=60 --json
? npm pkg delete <key>
Deletes a key from your package.json
The same syntax used to set values from your package can also be used
to remove existing ones. For example, in order to remove a script named
build:
npm pkg delete scripts.build

Workspaces support

You can set/get/delete items across your configured workspaces by using the workspace or workspaces config options.

For example, setting a funding value across all configured workspaces of a project:

```
npm pkg set funding=https://example.com --ws
```

When using npm pkg get to retrieve info from your configured workspaces, the returned result will be in a json format in which top level keys are the names of each workspace, the values of these keys will be the result values returned from each of the configured workspaces, e.g:

```
npm pkg get name version --ws
{
  "a": {
    "name": "a",
    "version": "1.0.0"
  },
  "b": {
    "name": "b",
    "version": "1.0.0"
  }
}
```

Configuration

force

? Default: false

? Type: Boolean

Removes various protections against unfortunate side effects, common mistakes, unnecessary performance degradation, and malicious input.

? Allow clobbering non-npm files in global installs.

? Allow the npm version command to work on an unclean git repository.

? Allow deleting the cache folder with npm cache clean.

? Allow installing packages that have an engines declaration requiring a different version of npm.

? Allow installing packages that have an engines declaration requiring

a different version of node, even if --engine-strict is enabled.

? Allow npm audit fix to install modules outside your stated dependency range (including SemVer-major changes).

? Allow unpublishing all versions of a published package.

? Allow conflicting peerDependencies to be installed in the root project.

? Implicitly set --yes during npm init.

? Allow clobbering existing values in npm pkg

? Allow unpublishing of entire packages (not just a single version).

If you don't have a clear idea of what you want to do, it is strongly recommended that you do not use this option!

json

? Default: false

? Type: Boolean

Whether or not to output JSON data, rather than the normal output.

? In npm pkg set it enables parsing set values with JSON.parse() before saving them to your package.json.

Not supported by all npm commands.

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 de? fined 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.

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.

See Also

? npm help install

? npm help init

? npm help config

? npm help set-script

? npm help workspaces

February 2023 NPM-PKG(1)