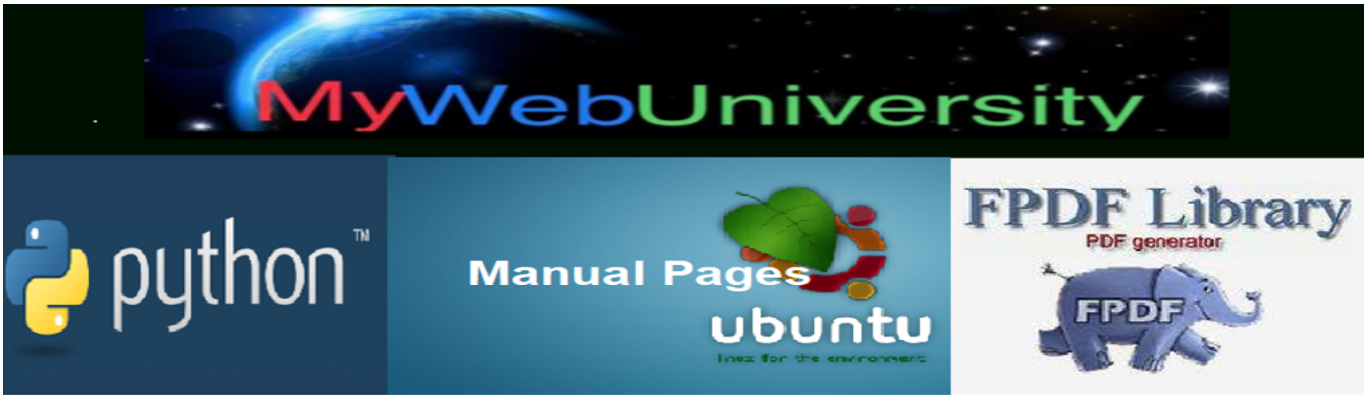

Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!

Rocky Enterprise Linux 9.2 Manual Pages on command 'npm-ls.1'

$ man npm-ls.1

NPM-LS(1) NPM-LS(1)

NAME

npm-ls - List installed packages

Synopsis

npm ls [[<@scope>/]<pkg> ...]

aliases: list, la, ll

Description

This command will print to stdout all the versions of packages that are installed, as well as their dependencies when --all is specified, in a tree structure.

Note: to get a "bottoms up" view of why a given package is included in the tree at all, use npm help explain.

Positional arguments are name@version-range identifiers, which will limit the results to only the paths to the packages named. Note that nested packages will also show the paths to the specified packages. For example, running npm ls promzard in npm's source tree will show:

npm@8.5.1 /path/to/npm

??? init-package-json@0.0.4

??? promzard@0.1.5

It will print out extraneous, missing, and invalid packages.

If a project specifies git urls for dependencies these are shown in parentheses after the name@version to make it easier for users to recognize potential forks of a project.

The tree shown is the logical dependency tree, based on package dependencies, not the physical layout of your node_modules folder.

When run as ll or la, it shows extended information by default.

Note: Design Changes Pending

The npm ls command's output and behavior made a ton of sense when npm created a node_mod?
ules folder that naively nested every dependency. In such a case, the logical dependency
graph and physical tree of packages on disk would be roughly identical.
With the advent of automatic install-time deduplication of dependencies in npm v3, the ls
output was modified to display the logical dependency graph as a tree structure, since
this was more useful to most users. However, without using npm ls -l, it became impossi?
ble show where a package was actually installed much of the time!
With the advent of automatic installation of peerDependencies in npm v7, this gets even
more curious, as peerDependencies are logically "underneath" their dependents in the de?
pendency graph, but are always physically at or above their location on disk.
Also, in the years since npm got an ls command (in version 0.0.2!), dependency graphs have
gotten much larger as a general rule. Therefore, in order to avoid dumping an excessive
amount of content to the terminal, npm ls now only shows the top level dependencies, un?
less --all is provided.
A thorough re-examination of the use cases, intention, behavior, and output of this com?
mand, is currently underway. Expect significant changes to at least the default hu?
man-readable npm ls output in npm v8.

Configuration

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

all

? Default: false

? Type: Boolean

When running npm outdated and npm ls, setting --all will show all outdated or installed
packages, rather than only those directly depended upon by the current project. <!-- au?
tomatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js
-->

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. <!-- automatically generated, do not edit manually -->

<!-- see lib/utils/config/definitions.js -->

long

? Default: false

? Type: Boolean

Show extended information in ls, search, and help-search. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

parseable

? Default: false

? Type: Boolean

Output parseable results from commands that write to standard output. For npm search, this will be tab-separated table format. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

global

? Default: false

? Type: Boolean

Operates in "global" mode, so that packages are installed into the prefix folder instead of the current working directory. See npm help folders for more on the differences in be? havior.

? packages are installed into the {prefix}/lib/node_modules folder, instead of the current working directory.

? bin files are linked to {prefix}/bin

? man pages are linked to {prefix}/share/man

<!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/defini? tions.js -->

depth

? Default: Infinity if --all is set, otherwise 1

? Type: null or Number

The depth to go when recursing packages for npm ls.

If not set, npm ls will show only the immediate dependencies of the root project. If --all is set, then npm will show all dependencies by default. <!-- automatically generated, do

not edit manually --> <!-- see lib/utils/config/definitions.js -->

omit

? Default: 'dev' if the NODE_ENV environment variable is set to 'production', otherwise empty.

? Type: "dev", "optional", or "peer" (can be set multiple times)

Dependency types to omit from the installation tree on disk.

Note that these dependencies are still resolved and added to the package-lock.json or npm-shrinkwrap.json file. They are just not physically installed on disk.

If a package type appears in both the --include and --omit lists, then it will be in? cluded.

If the resulting omit list includes 'dev', then the NODE_ENV environment variable will be set to 'production' for all lifecycle scripts. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

link

? Default: false

? Type: Boolean

Used with npm ls, limiting output to only those packages that are linked. <!-- automati? cally generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

package-lock-only

? Default: false

? Type: Boolean

If set to true, the current operation will only use the package-lock.json, ignoring node_modules.

For update this means only the package-lock.json will be updated, instead of checking node_modules and downloading dependencies.

For list this means the output will be based on the tree described by the pack? age-lock.json, rather than the contents of node_modules. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

unicode

? Default: false on windows, true on mac/unix systems with a unicode locale, as defined by the LC_ALL, LC_CTYPE, or LANG environment variables.

? Type: Boolean

When set to true, npm uses unicode characters in the tree output. When false, it uses

ascii characters instead of unicode glyphs. <!-- 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 op? tion.

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 al? together. 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 spec? ified 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 explain

? npm help config

? npm help npmrc

? npm help folders

? npm help explain

? npm help install

? npm help link

? npm help prune

? npm help outdated

? npm help update