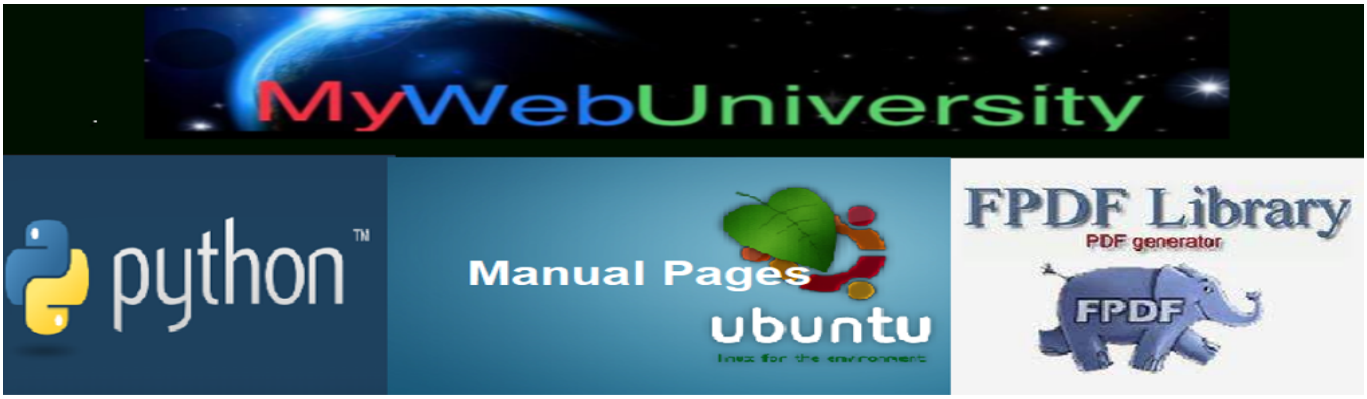

Rocky Enterprise Linux 9.2 Manual Pages on command 'npm-dedupe.1'

$ man npm-dedupe.1

NPM-DEDUPE(1) NPM-DEDUPE(1)

NAME

npm-dedupe - Reduce duplication in the package tree

Synopsis

npm dedupe

npm ddp

aliases: ddp

Description

Searches the local package tree and attempts to simplify the overall structure by moving dependencies further up the tree, where they can be more effectively shared by multiple dependent packages.

For example, consider this dependency graph:

```
a
+-- b <-- depends on c@1.0.x
|   `-- c@1.0.3
`-- d <-- depends on c@~1.0.9
    `-- c@1.0.10
```

In this case, npm dedupe will transform the tree to:

```
a
+-- b
+-- d
`-- c@1.0.10
```

Because of the hierarchical nature of node's module lookup, b and d will both get their

dependency met by the single c package at the root level of the tree.

In some cases, you may have a dependency graph like this:

```
a
+-- b <-- depends on c@1.0.x
+-- c@1.0.3
`-- d <-- depends on c@1.x
    `-- c@1.9.9
```

During the installation process, the c@1.0.3 dependency for b was placed in the root of the tree. Though d's dependency on c@1.x could have been satisfied by c@1.0.3, the newer c@1.9.0 dependency was used, because npm favors updates by default, even when doing so causes duplication.

Running npm dedupe will cause npm to note the duplication and re-evaluate, deleting the nested c module, because the one in the root is sufficient.

To prefer deduplication over novelty during the installation process, run npm install --prefer-dedupe or npm config set prefer-dedupe true.

Arguments are ignored. Dedupe always acts on the entire tree.

Note that this operation transforms the dependency tree, but will never result in new mod? ules being installed.

Using npm find-dupes will run the command in --dry-run mode.

Note that by default npm dedupe will not update the semver values of direct dependencies in your project package.json, if you want to also update values in package.json you can run: npm dedupe --save (or add the save=true option to a npm help configuration file to make that the default behavior).

Configuration

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

global-style

? Default: false

? Type: Boolean

Causes npm to install the package into your local node_modules folder with the same layout it uses with the global node_modules folder. Only your direct dependencies will show in node_modules and everything they depend on will be flattened in their node_modules fold? ers. This obviously will eliminate some deduping. If used with legacy-bundling,

legacy-bundling will be preferred. <!-- automatically generated, do not edit manually -->

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

legacy-bundling

? Default: false

? Type: Boolean

Causes npm to install the package such that versions of npm prior to 1.4, such as the one included with node 0.8, can install the package. This eliminates all automatic deduping. If used with global-style this option will be preferred. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

strict-peer-deps

? Default: false

? Type: Boolean

If set to true, and --legacy-peer-deps is not set, then any conflicting peerDependencies will be treated as an install failure, even if npm could reasonably guess the appropriate resolution based on non-peer dependency relationships.

By default, conflicting peerDependencies deep in the dependency graph will be resolved us? ing the nearest non-peer dependency specification, even if doing so will result in some packages receiving a peer dependency outside the range set in their package's peerDepen? dencies object.

When such and override is performed, a warning is printed, explaining the conflict and the packages involved. If --strict-peer-deps is set, then this warning is treated as a fail? ure. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/con? fig/definitions.js -->

package-lock

? Default: true

? Type: Boolean

If set to false, then ignore package-lock.json files when installing. This will also pre? vent writing package-lock.json if save is true.

When package package-locks are disabled, automatic pruning of extraneous modules will also be disabled. To remove extraneous modules with package-locks disabled use npm prune. <!-- 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 -->

ignore-scripts

? Default: false

? Type: Boolean

If true, npm does not run scripts specified in package.json files.

Note that commands explicitly intended to run a particular script, such as npm start, npm stop, npm restart, npm test, and npm run-script will still run their intended script if ignore-scripts is set, but they will not run any pre- or post-scripts. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

audit

? Default: true

? Type: Boolean

When "true" submit audit reports alongside the current npm command to the default registry and all registries configured for scopes. See the documentation for npm help audit for de? tails on what is submitted. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

bin-links

? Default: true

? Type: Boolean

Tells npm to create symlinks (or .cmd shims on Windows) for package executables.

Set to false to have it not do this. This can be used to work around the fact that some file systems don't support symlinks, even on ostensibly Unix systems. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

fund

? Default: true

? Type: Boolean

When "true" displays the message at the end of each npm install acknowledging the number of dependencies looking for funding. See npm help npm fund for details. <!-- automati? cally generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

dry-run

? Default: false

? Type: Boolean

Indicates that you don't want npm to make any changes and that it should only report what it would have done. This can be passed into any of the commands that modify your local in? stallation, eg, install, update, dedupe, uninstall, as well as pack and publish.

Note: This is NOT honored by other network related commands, eg dist-tags, owner, etc. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/defini? tions.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 find-dupes

? npm help ls

? npm help update

? npm help install

undefined NaN NPM-DEDUPE(1)