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

$ man npm-dedupe.1

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

NAME

   npm-dedupe - Reduce duplication in the package tree

 Synopsis

    npm dedupe

    alias: 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 dupli?cation.

Running npm dedupe will cause npm to note the duplication and re-evalu?ate, deleting the nested c module, because the one in the root is suf?ficient.

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 modules being installed.

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

Note: npm dedupe will never update the semver values of direct depen?dencies in your project package.json, if you want to update values in package.json you can run: npm update --save instead.

Configuration

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 folders. This obvi?ously will eliminate some deduping. If used with legacy-bundling,

legacy-bundling will be preferred.

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.

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 de? pendency relationships.

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

package-lock

? Default: true

? Type: Boolean

If set to false, then ignore package-lock.json files when installing. This will also prevent writing package-lock.json if save is true. This configuration does not affect npm ci.

omit

? Default: 'dev' if the NODE_ENV environment variable is set to 'pro? duction', 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 pack?

age-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 included.

If the resulting omit list includes 'dev', then the NODE_ENV environ?

ment variable will be set to 'production' for all lifecycle scripts.

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.

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 details on what is submitted.

bin-links

? Default: true

? Type: Boolean

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

ecutables.

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 osten?

sibly Unix systems.

fund

? Default: true

? Type: Boolean

When "true" displays the message at the end of each npm install ac?

knowledging the number of dependencies looking for funding. See npm
help fund for details.

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 installation, eg, install, up?
date, dedupe, uninstall, as well as pack and publish.

Note: This is NOT honored by other network related commands, eg
dist-tags, owner, etc.

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.

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.

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

install-links

? Default: false

? Type: Boolean

When set file: protocol dependencies that exist outside of the project root will be packed and installed as regular dependencies instead of creating a symlink. This option has no effect on workspaces.

See Also

? npm help find-dupes

? npm help ls

? npm help update

? npm help install