



Red Hat Enterprise Linux Release 9.2 Manual Pages on 'dnf.8' command

\$ man dnf.8

DNF(8) DNF DNF(8)

NAME

dnf - DNF Command Reference

SYNOPSIS

dnf [options] <command> [<args>...]

DESCRIPTION

DNF is the next upcoming major version of YUM, a package manager for RPM-based Linux distributions. It roughly maintains CLI compatibility with YUM and defines a strict API for extensions and plugins.

Plugins can modify or extend features of DNF or provide additional CLI commands on top of those mentioned below. If you know the name of such a command (including commands mentioned below), you may find/install the package which provides it using the appropriate virtual provide in the form of `dnf-command(<alias>)`, where `<alias>` is the name of the command; e.g. `dnf install 'dnf-command(versionlock)'` installs a versionlock plugin. This approach also applies to specifying dependencies of packages that require a particular DNF command.`

Return values:

- ? 0 : Operation was successful.
- ? 1 : An error occurred, which was handled by dnf.
- ? 3 : An unknown unhandled error occurred during operation.
- ? 100: See check-update
- ? 200: There was a problem with acquiring or releasing of locks.

Available commands:

? alias
? autoremove
? check
? check-update
? clean
? deplist
? distro-sync
? downgrade
? group
? help
? history
? info
? install
? list
? makecache
? mark
? module
? provides
? reinstall
? remove
? repoinfo
? repolist
? repoquery
? repository-packages
? search
? shell
? swap
? updateinfo
? upgrade
? upgrade-minimal

Additional information:

? Options

- ? Specifying Packages
- ? Specifying Provides
- ? Specifying File Provides
- ? Specifying Groups
- ? Specifying Transactions
- ? Metadata Synchronization
- ? Configuration Files Replacement Policy
- ? Files
- ? See Also

OPTIONS

-4 Resolve to IPv4 addresses only.

-6 Resolve to IPv6 addresses only.

--advisory=<advisory>, --advisories=<advisory>

Include packages corresponding to the advisory ID, Eg. FE? DORA-2201-123. Applicable for the install, repoquery, update? info, upgrade and offline-upgrade (dnf-plugins-core) commands.

--allowerasing

Allow erasing of installed packages to resolve dependencies. This option could be used as an alternative to the yum swap command where packages to remove are not explicitly defined.

--assumeno

Automatically answer no for all questions.

-b, --best

Try the best available package versions in transactions. Specifically during dnf upgrade, which by default skips over updates that can not be installed for dependency reasons, the switch forces DNF to only consider the latest packages. When running into packages with broken dependencies, DNF will fail giving a reason why the latest version can not be installed.

Note that the use of the newest available version is only guaranteed for the packages directly requested (e.g. as a command line arguments), and the solver may use older versions of dependencies to meet their requirements.

`--bugfix`

Include packages that fix a bugfix issue. Applicable for the `install`, `repoquery`, `updateinfo`, `upgrade` and `offline-upgrade` (`dnf-plugins-core`) commands.

`--bz=<bugzilla>`, `--bzs=<bugzilla>`

Include packages that fix a Bugzilla ID, Eg. 123123. Applicable for the `install`, `repoquery`, `updateinfo`, `upgrade` and `offline-upgrade` (`dnf-plugins-core`) commands.

`-C`, `--cacheonly`

Run entirely from system cache, don't update the cache and use it even in case it is expired.

DNF uses a separate cache for each user under which it executes.

The cache for the root user is called the system cache. This switch allows a regular user read-only access to the system cache, which usually is more fresh than the user's and thus he does not have to wait for metadata sync.

`--color=<color>`

Control whether color is used in terminal output. Valid values are `always`, `never` and `auto` (default).

`--comment=<comment>`

Add a comment to the transaction history.

`-c <config file>`, `--config=<config file>`

Configuration file location.

`--cve=<cves>`, `--cves=<cves>`

Include packages that fix a CVE (Common Vulnerabilities and Exposures) ID (<http://cve.mitre.org/about/>), Eg. CVE-2201-0123.

Applicable for the `install`, `repoquery`, `updateinfo`, `upgrade` and `offline-upgrade` (`dnf-plugins-core`) commands.

`-d <debug level>`, `--debuglevel=<debug level>`

Debugging output level. This is an integer value between 0 (no additional information strings) and 10 (shows all debugging information, even that not understandable to the user), default is 2. Deprecated, use `-v` instead.

`--debugsolver`

Dump data aiding in dependency solver debugging into `./debug?` data.

`--disableexcludes=[all|main|<repoid>],` `--disableexcludep?`

`kgs=[all|main|<repoid>]`

Disable the configuration file excludes. Takes one of the following three options:

? all, disables all configuration file excludes

? main, disables excludes defined in the [main] section

? repoid, disables excludes defined for the given repository

`--disable, --set-disabled`

Disable specified repositories (automatically saves). The option has to be used together with the `config-manager` command (`dnf-plugins-core`).

`--disableplugin=<plugin names>`

Disable the listed plugins specified by names or globs.

`--disablerepo=<repoid>`

Temporarily disable active repositories for the purpose of the current `dnf` command. Accepts an id, a comma-separated list of ids, or a glob of ids. This option can be specified multiple times, but is mutually exclusive with `--repo`.

`--downloadaddir=<path>, --destdir=<path>`

Redirect downloaded packages to provided directory. The option has to be used together with the `--downloadonly` command line option, with the `download`, `modulesync`, `reposync` or `system-upgrade` commands (`dnf-plugins-core`).

`--downloadonly`

Download the resolved package set without performing any rpm transaction (install/upgrade/erase).

Packages are removed after the next successful transaction. This applies also when used together with `--destdir` option as the directory is considered as a part of the DNF cache. To persist the packages, use the `download` command instead.

`-e <error level>, --errorlevel=<error level>`

Error output level. This is an integer value between 0 (no error output) and 10 (shows all error messages), default is 3. Deprecated, use `-v` instead.

`--enable, --set-enabled`

Enable specified repositories (automatically saves). The option has to be used together with the `config-manager` command (`dnf-plugins-core`).

`--enableplugin=<plugin names>`

Enable the listed plugins specified by names or globs.

`--enablerepo=<repoid>`

Temporarily enable additional repositories for the purpose of the current `dnf` command. Accepts an id, a comma-separated list of ids, or a glob of ids. This option can be specified multiple times.

`--enhancement`

Include enhancement relevant packages. Applicable for the `install`, `repoquery`, `updateinfo`, `upgrade` and `offline-upgrade` (`dnf-plugins-core`) commands.

`-x <package-file-spec>, --exclude=<package-file-spec>`

Exclude packages specified by `<package-file-spec>` from the operation.

`--excludepkgs=<package-file-spec>`

Deprecated option. It was replaced by the `--exclude` option.

`--forcearch=<arch>`

Force the use of an architecture. Any architecture can be specified. However, use of an architecture not supported natively by your CPU will require emulation of some kind. This is usually through QEMU. The behavior of `--forcearch` can be configured by using the `arch` and `ignorearch` configuration options with values `<arch>` and `True` respectively.

`-h, --help, --help-cmd`

Show the help.

`--installroot=<path>`

Specifies an alternative `installroot`, relative to where all packages will be installed. Think of this like doing `chroot <root> dnf`, except using `--installroot` allows `dnf` to work before the `chroot` is created. It requires absolute path.

? `cachedir`, `log files`, `releasever`, and `gpgkey` are taken from or stored in the `installroot`. `Gpgkeys` are imported into the `installroot` from a path relative to the host which can be specified in the repository section of configuration files.

? configuration file and `reposdir` are searched inside the `installroot` first. If they are not present, they are taken from the host system.

Note: When a path is specified within a command line argument (`--config=<config file>` in case of configuration file and `--setopt=reposdir=<reposdir>` for `reposdir`) then this path is always relative to the host with no exceptions.

? `vars` are taken from the host system or `installroot` according to `reposdir`. When `reposdir` path is specified within a command line argument, `vars` are taken from the `installroot`. When `varsdir` paths are specified within a command line argument (`--setopt=varsdir=<reposdir>`) then those path are always relative to the host with no exceptions.

? The `pluginpath` and `pluginconfpath` are relative to the host.

Note: You may also want to use the command-line option `--releasever=<release>` when creating the `installroot`, otherwise the `$releasever` value is taken from the `rpmdb` within the `installroot` (and thus it is empty at the time of creation and the transaction will fail).

If `--releasever=/` is used, the `releasever` will be detected from the host (`/`) system. The new `installroot` path at the time of creation does not contain the repository, `releasever` and `dnf.conf` files.

On a modular system you may also want to use the `--setopt=module_platform_id=<module_platform_name:stream>` command-line option when creating the `installroot`, otherwise the `module_platform_id` value will be taken from the `/etc/os-release` file within the in?

installroot (and thus it will be empty at the time of creation, the modular dependency could be unsatisfied and modules content could be excluded).

Installroot examples:

`dnf --installroot=<installroot> --releasever=<release> install sys?`

`tem-release`

Permanently sets the releasever of the system in the <installroot> directory to <release>.

`dnf --installroot=<installroot> --setopt=reposdir=<path> --config /path/dnf.conf upgrade`

Upgrades packages inside the installroot from a repository described by --setopt using configuration from /path/dnf.conf.

`--newpackage`

Include newpackage relevant packages. Applicable for the install, repoquery, updateinfo, upgrade and offline-upgrade (dnf-plugins-core) commands.

`--noautoremove`

Disable removal of dependencies that are no longer used. It sets clean_requirements_on_remove configuration option to False.

`--nobest`

Set best option to False, so that transactions are not limited to best candidates only.

`--nodoc`

Do not install documentation. Sets the rpm flag 'RPMTRANS_FLAG_NODOCS'.

`--nogpgcheck`

Skip checking GPG signatures on packages (if RPM policy allows).

`--nopugins`

Disable all plugins.

`--obsoletes`

This option has an effect on an install/update, it enables dnf's obsoletes processing logic. For more information see the obso?

letes option.

This option also displays capabilities that the package obso?

letes when used together with the repoquery command.

Configuration Option: obsoletes

`-q, --quiet`

In combination with a non-interactive command, shows just the relevant content. Suppresses messages notifying about the current state or actions of DNF.

`-R <minutes>, --randomwait=<minutes>`

Maximum command wait time.

`--refresh`

Set metadata as expired before running the command.

`--releasever=<release>`

Configure DNF as if the distribution release was <release>. This can affect cache paths, values in configuration files and mirrorlist URLs.

`--repofrompath <repo>,<path/url>`

Specify a repository to add to the repositories for this query.

This option can be used multiple times.

? The repository label is specified by <repo>.

? The path or url to the repository is specified by <path/url>. It is the same path as a baseurl and can be also enriched by the repository variables.

? The configuration for the repository can be adjusted using -?

`-setopt=<repo>.<option>=<value>.`

? If you want to view only packages from this repository, combine this with the `--repo=<repo>` or `--disablerepo="*"` switches.

`--repo=<repoid>, --repoid=<repoid>`

Enable just specific repositories by an id or a glob. Can be used multiple times with accumulative effect. It is basically a shortcut for `--disablerepo="*" --enablerepo=<repoid>` and is mutually exclusive with the `--disablerepo` option.

`--rpmverbosity=<name>`

RPM debug scriptlet output level. Sets the debug level to <name> for RPM scriptlets. For available levels, see the `rpmverbosity` configuration option.

`--sec-severity=<severity>`, `--secseverity=<severity>`

Includes packages that provide a fix for an issue of the specified severity. Applicable for the `install`, `repoquery`, `updateinfo`, `upgrade` and `offline-upgrade` (`dnf-plugins-core`) commands.

`--security`

Includes packages that provide a fix for a security issue. Applicable for the `install`, `repoquery`, `updateinfo`, `upgrade` and `offline-upgrade` (`dnf-plugins-core`) commands.

`--setopt=<option>=<value>`

Override a configuration option from the configuration file. To override configuration options for repositories, use `repoid.option` for the <option>. Values for configuration options like `excludepkgs`, `includepkgs`, `installonlypkgs` and `tsflags` are appended to the original value, they do not override it. However, specifying an empty value (e.g. `--setopt=tsflags=`) will clear the option.

`--skip-broken`

Resolve `depsolve` problems by removing packages that are causing problems from the transaction. It is an alias for the `strict` configuration option with value `False`. Additionally, with the `enable` and `disable` module subcommands it allows one to perform an action even in case of broken modular dependencies.

`--showduplicates`

Show duplicate packages in repositories. Applicable for the `list` and `search` commands.

`-v`, `--verbose`

Verbose operation, show debug messages.

`--version`

Show DNF version and exit.

`-y`, `--assumeyes`

Automatically answer yes for all questions.

List options are comma-separated. Command-line options override respective settings from configuration files.

COMMANDS

For an explanation of <package-spec>, <package-file-spec> and <package-name-spec> see Specifying Packages.

For an explanation of <provide-spec> see Specifying Provides.

For an explanation of <group-spec> see Specifying Groups.

For an explanation of <module-spec> see Specifying Modules.

For an explanation of <transaction-spec> see Specifying Transactions.

Alias Command

Command: alias

Allows the user to define and manage a list of aliases (in the form <name=value>), which can be then used as dnf commands to abbreviate longer command sequences. For examples on using the alias command, see Alias Examples. For examples on the alias processing, see Alias Processing Examples.

To use an alias (name=value), the name must be placed as the first "command" (e.g. the first argument that is not an option). It is then replaced by its value and the resulting sequence is again searched for aliases. The alias processing stops when the first found command is not a name of any alias.

In case the processing would result in an infinite recursion, the original arguments are used instead.

Also, like in shell aliases, if the result starts with a \, the alias processing will stop.

All aliases are defined in configuration files in the /etc/dnf/aliases.d/ directory in the [aliases] section, and aliases created by the alias command are written to the USER.conf file. In case of conflicts, the USER.conf has the highest priority, and alphabetical ordering is used for the rest of the configuration files.

Optionally, there is the enabled option in the [main] section defaulting to True. This can be set for each file separately in the respective

file, or globally for all aliases in the ALIASES.conf file.

```
dnf alias [options] [list] [<name>...]
```

List aliases with their final result. The [<alias>...] parameter further limits the result to only those aliases matching it.

```
dnf alias [options] add <name=value>...
```

Create new aliases.

```
dnf alias [options] delete <name>...
```

Delete aliases.

Alias Examples

```
dnf alias list
```

Lists all defined aliases.

```
dnf alias add rm=remove
```

Adds a new command alias called rm which works the same as the remove command.

```
dnf alias add upgrade="\upgrade --skip-broken --disableexcludes=all  
--obsoletes"
```

Adds a new command alias called upgrade which works the same as the upgrade command, with additional options. Note that the original upgrade command is prefixed with a \ to prevent an infinite loop in alias processing.

Alias Processing Examples

If there are defined aliases in=install and FORCE="--skip-broken --disableexcludes=all":

? dnf FORCE in will be replaced with dnf --skip-broken --disableexcludes=all install

? dnf in FORCE will be replaced with dnf install FORCE (which will fail)

If there is defined alias in=install:

? dnf in will be replaced with dnf install

? dnf --repo updates in will be replaced with dnf --repo updates in (which will fail)

Autoremove Command

Command: autoremove

Aliases for explicit NEVRA matching: autoremove-n, autoremove-na, autoremove-nevra

`dnf [options] autoremove`

Removes all "leaf" packages from the system that were originally installed as dependencies of user-installed packages, but which are no longer required by any such package.

Packages listed in `installonlypkgs` are never automatically removed by this command.

`dnf [options] autoremove <spec>...`

This is an alias for the `Remove Command` with `clean_requirements_on_remove` set to `True`. It removes the specified packages from the system along with any packages depending on the packages being removed. Each `<spec>` can be either a `<package-spec>`, which specifies a package directly, or a `@<group-spec>`, which specifies an (environment) group which contains it. It also removes any dependencies that are no longer needed.

There are also a few specific autoremove commands `autoremove-n`, `autoremove-na` and `autoremove-nevra` that allow the specification of an exact argument in the NEVRA (name-epoch:version-release.architecture) format.

This command by default does not force a sync of expired metadata. See also [Metadata Synchronization](#).

Check Command

Command: `check`

`dnf [options] check [--dependencies] [--duplicates] [--obsoleted] [--provides]`

Checks the local `packagedb` and produces information on any problems it finds. You can limit the checks to be performed by using the `--dependencies`, `--duplicates`, `--obsoleted` and `--provides` options (the default is to check everything).

Check-Update Command

Command: `check-update`

Aliases: `check-upgrade`

`dnf [options] check-update [--changelogs] [<package-file-spec>...]`

Non-interactively checks if updates of the specified packages are available. If no <package-file-spec> is given, checks whether any updates at all are available for your system. DNF exit code will be 100 when there are updates available and a list of the updates will be printed, 0 if not and 1 if an error occurs. If --changelogs option is specified, also changelog delta of packages about to be updated is printed.

Please note that having a specific newer version available for an installed package (and reported by check-update) does not imply that subsequent dnf upgrade will install it. The difference is that dnf upgrade has restrictions (like package dependencies being satisfied) to take into account.

The output is affected by the autocheck_running_kernel configuration option.

Clean Command

Command: clean

Performs cleanup of temporary files kept for repositories. This includes any such data left behind from disabled or removed repositories as well as for different distribution release versions.

`dnf clean dbcache`

Removes cache files generated from the repository metadata. This forces DNF to regenerate the cache files the next time it is run.

`dnf clean expire-cache`

Marks the repository metadata expired. DNF will re-validate the cache for each repository the next time it is used.

`dnf clean metadata`

Removes repository metadata. Those are the files which DNF uses to determine the remote availability of packages. Using this option will make DNF download all the metadata the next time it is run.

`dnf clean packages`

Removes any cached packages from the system.

dnf clean all

Does all of the above.

Deplist Command

dnf [options] deplist [<select-options>] [<query-options>] [<package-spec>]

Deprecated alias for dnf repoquery --deplist.

Distro-Sync Command

Command: distro-sync

Aliases: dsync

Deprecated aliases: distrosync, distribution-synchronization

dnf distro-sync [<package-spec>...]

As necessary upgrades, downgrades or keeps selected installed packages to match the latest version available from any enabled repository. If no package is given, all installed packages are considered.

See also Configuration Files Replacement Policy.

Downgrade Command

Command: downgrade

Aliases: dg

dnf [options] downgrade <package-spec>...

Downgrades the specified packages to the highest installable package of all known lower versions if possible. When version is given and is lower than version of installed package then it downgrades to target version.

Group Command

Command: group

Aliases: grp

Deprecated aliases: groups, grouplist, groupinstall, groupupdate, groupremove, grouperase, groupinfo

Groups are virtual collections of packages. DNF keeps track of groups that the user selected ("marked") installed and can manipulate the comprising packages with simple commands.

dnf [options] group [summary] <group-spec>

Display overview of how many groups are installed and available.

With a spec, limit the output to the matching groups. summary is the default groups subcommand.

`dnf [options] group info <group-spec>`

Display package lists of a group. Shows which packages are in? stalled or available from a repository when -v is used.

`dnf [options] group install [--with-optional] <group-spec>...`

Mark the specified group installed and install packages it contains. Also include optional packages of the group if --with-optional is specified. All mandatory and Default packages will be installed whenever possible. Conditional packages are installed if they meet their requirement. If the group is already (partially) installed, the command installs the missing packages from the group. Depending on the value of obsoletes configuration option group installation takes obsoletes into account.

`dnf [options] group list <group-spec>...`

List all matching groups, either among installed or available groups. If nothing is specified, list all known groups. --installed and --available options narrow down the requested list. Records are ordered by the display_order tag defined in comps.xml file. Provides a list of all hidden groups by using option --hidden. Provides group IDs when the -v or --ids options are used.

`dnf [options] group remove <group-spec>...`

Mark the group removed and remove those packages in the group from the system which do not belong to another installed group and were not installed explicitly by the user.

`dnf [options] group upgrade <group-spec>...`

Upgrades the packages from the group and upgrades the group itself. The latter comprises of installing packages that were added to the group by the distribution and removing packages that got removed from the group as far as they were not installed explicitly by the user.

Groups can also be marked installed or removed without physically ma?

manipulating any packages:

`dnf [options] group mark install <group-spec>...`

Mark the specified group installed. No packages will be installed by this command, but the group is then considered installed.

`dnf [options] group mark remove <group-spec>...`

Mark the specified group removed. No packages will be removed by this command.

See also Configuration Files Replacement Policy.

Help Command

Command: `help`

`dnf help [<command>]`

Displays the help text for all commands. If given a command name then only displays help for that particular command.

History Command

Command: `history`

Aliases: `hist`

The history command allows the user to view what has happened in past transactions and act according to this information (assuming the `history_record` configuration option is set).

`dnf history [list] [--reverse] [<spec>...]`

The default history action is listing information about given transactions in a table. Each `<spec>` can be either a `<transaction-spec>`, which specifies a transaction directly, or a `<transaction-spec>..<transaction-spec>, which specifies a range of transactions, or a <package-name-spec>, which specifies a transaction by a package which it manipulated. When no transaction is specified, list all known transactions.`

`--reverse`

The order of history list output is printed in reverse order.

`dnf history info [<spec>...]`

Describe the given transactions. The meaning of `<spec>` is the

same as in the History List Command. When no transaction is specified, describe what happened during the latest transaction.

`dnf history redo <transaction-spec>|<package-file-spec>`

Repeat the specified transaction. Uses the last transaction (with the highest ID) if more than one transaction for given <package-file-spec> is found. If it is not possible to redo some operations due to the current state of RPMDB, it will not redo the transaction.

`dnf history replay [--ignore-installed] [--ignore-extras] [--skip-unavailable] <filename>`

Replay a transaction stored in file <filename> by History Store Command. The replay will perform the exact same operations on the packages as in the original transaction and will return with an error in case of any differences in installed packages or their versions. See also the Transaction JSON Format specification of the file format.

`--ignore-installed`

Don't check for the installed packages being in the same state as those recorded in the transaction. E.g. in case there is an upgrade `foo-1.0 -> foo-2.0` stored in the transaction, but there is `foo-1.1` installed on the target system.

`--ignore-extras`

Don't check for extra packages pulled into the transaction on the target system. E.g. the target system may not have some dependency, which was installed on the source system. The replay errors out on this by default, as the transaction would not be the same.

`--skip-unavailable`

In case some packages stored in the transaction are not available on the target system, skip them instead of erroring out.

`dnf history rollback <transaction-spec>|<package-file-spec>`

Undo all transactions performed after the specified transaction.

Uses the last transaction (with the highest ID) if more than one transaction for given <package-file-spec> is found. If it is not possible to undo some transactions due to the current state of RPMDB, it will not undo any transaction.

`dnf history store [--output <output-file>] <transaction-spec>`

Store a transaction specified by <transaction-spec>. The transaction can later be replayed by the History Replay Command.

Warning: The stored transaction format is considered unstable and may change at any time. It will work if the same version of dnf is used to store and replay (or between versions as long as it stays the same).

`-o <output-file>, --output=<output-file>` Store the serialized transaction into <output-file>. Default is transaction.json.

`dnf history undo <transaction-spec>|<package-file-spec>`

Perform the opposite operation to all operations performed in the specified transaction. Uses the last transaction (with the highest ID) if more than one transaction for given <package-file-spec> is found. If it is not possible to undo some operations due to the current state of RPMDB, it will not undo the transaction.

`dnf history userinstalled`

Show all installonly packages, packages installed outside of DNF and packages not installed as dependency. I.e. it lists packages that will stay on the system when Autoremove Command or Remove Command along with `clean_requirements_on_remove` configuration option set to True is executed. Note the same results can be accomplished with `dnf repoquery --userinstalled`, and the `repoquery` command is more powerful in formatting of the output.

This command by default does not force a sync of expired metadata, except for the redo, rollback, and undo subcommands. See also Metadata Synchronization and Configuration Files Replacement Policy.

Command: info

Aliases: if

`dnf [options] info [<package-file-spec>...]`

Lists description and summary information about installed and available packages.

The info command limits the displayed packages the same way as the list command.

This command by default does not force a sync of expired metadata. See also Metadata Synchronization.

Install Command

Command: install

Aliases: in

Aliases for explicit NEVRA matching: install-n, install-na, install-nevra

Deprecated aliases: localinstall

`dnf [options] install <spec>...`

Makes sure that the given packages and their dependencies are installed on the system. Each <spec> can be either a <package-spec>, or a @<module-spec>, or a @<group-spec>. See Install Examples. If a given package or provide cannot be (and is not already) installed, the exit code will be non-zero. If the <spec> matches both a @<module-spec> and a @<group-spec>, only the module is installed.

When <package-spec> to specify the exact version of the package is given, DNF will install the desired version, no matter which version of the package is already installed. The former version of the package will be removed in the case of non-installonly package.

On the other hand if <package-spec> specifies only a name, DNF also takes into account packages obsoleting it when picking which package to install. This behaviour is specific to the install command. Note that this can lead to seemingly unexpected results if a package has multiple versions and some older version is being obsoleted. It creates a split in the upgrade-path

and both ways are considered correct, the resulting package is picked simply by lexicographical order.

There are also a few specific install commands `install-n`, `install-na` and `install-nevra` that allow the specification of an exact argument in the NEVRA format.

See also Configuration Files Replacement Policy.

Install Examples

```
dnf install tit0
```

Install the tit0 package (tit0 is the package name).

```
dnf install ~/Downloads/tit0-0.6.2-1.fc22.noarch.rpm
```

Install a local rpm file tit0-0.6.2-1.fc22.noarch.rpm from the ~/Downloads/ directory.

```
dnf install tit0-0.5.6-1.fc22
```

Install the package with a specific version. If the package is already installed it will automatically try to downgrade or upgrade to the specific version.

```
dnf --best install tit0
```

Install the latest available version of the package. If the package is already installed it will try to automatically upgrade to the latest version. If the latest version of the package cannot be installed, the installation will fail.

```
dnf install vim
```

DNF will automatically recognize that vim is not a package name, but will look up and install a package that provides vim with all the required dependencies. Note: Package name match has precedence over package provides match.

```
dnf install https://kojipkgs.fedoraproject.org/packages/tit0/0.6.0/1.fc22/noarch/tit0-0.6.0-1.fc22.noarch.rpm
```

Install a package directly from a URL.

```
dnf install '@docker'
```

Install all default profiles of module 'docker' and their RPMs.

Module streams get enabled accordingly.

```
dnf install '@Web Server'
```

Install the 'Web Server' environmental group.

```
dnf install /usr/bin/rpmsign
```

Install a package that provides the /usr/bin/rpmsign file.

```
dnf -y install tito --setopt=install_weak_deps=False
```

Install the tito package (tito is the package name) without weak deps. Weak deps are not required for core functionality of the package, but they enhance the original package (like extended documentation, plugins, additional functions, etc.).

```
dnf install --advisory=FEDORA-2018-b7b99fe852 \*
```

Install all packages that belong to the "FEDORA-2018-b7b99fe852" advisory.

List Command

Command: list

Aliases: ls

Prints lists of packages depending on the packages' relation to the system. A package is installed if it is present in the RPMDB, and it is available if it is not installed but is present in a repository that DNF knows about.

The list command also limits the displayed packages according to specific criteria, e.g. to only those that update an installed package (respecting the repository priority). The exclude option in the configuration file can influence the result, but if the --disableexcludes command line option is used, it ensures that all installed packages will be listed.

```
dnf [options] list [--all] [<package-file-spec>...]
```

Lists all packages, present in the RPMDB, in a repository or both.

```
dnf [options] list --installed [<package-file-spec>...]
```

Lists installed packages.

```
dnf [options] list --available [<package-file-spec>...]
```

Lists available packages.

```
dnf [options] list --extras [<package-file-spec>...]
```

Lists extras, that is packages installed on the system that are

not available in any known repository.

```
dnf [options] list --obsoletes [<package-file-spec>...]
```

List packages installed on the system that are obsoleted by packages in any known repository.

```
dnf [options] list --recent [<package-file-spec>...]
```

List packages recently added into the repositories.

```
dnf [options] list --upgrades [<package-file-spec>...]
```

List upgrades available for the installed packages.

```
dnf [options] list --autoremove
```

List packages which will be removed by the dnf autoremove command.

This command by default does not force a sync of expired metadata. See also Metadata Synchronization.

Makecache Command

Command: makecache

Aliases: mc

```
dnf [options] makecache
```

Downloads and caches metadata for enabled repositories. Tries to avoid downloading whenever possible (e.g. when the local metadata hasn't expired yet or when the metadata timestamp hasn't changed).

```
dnf [options] makecache --timer
```

Like plain makecache, but instructs DNF to be more resource-aware, meaning it will not do anything if running on battery power and will terminate immediately if it's too soon after the last successful makecache run (see dnf.conf(5), metadata_timer_sync).

Mark Command

Command: mark

```
dnf mark install <package-spec>...
```

Marks the specified packages as installed by user. This can be useful if any package was installed as a dependency and is desired to stay on the system when Autoremove Command or Remove

Command along with `clean_requirements_on_remove` configuration option set to `True` is executed.

`dnf mark remove <package-spec>...`

Unmarks the specified packages as installed by user. Whenever you as a user don't need a specific package you can mark it for removal. The package stays installed on the system but will be removed when `Autoremove` Command or `Remove` Command along with `clean_requirements_on_remove` configuration option set to `True` is executed. You should use this operation instead of `Remove Command` if you're not sure whether the package is a requirement of other user installed packages on the system.

`dnf mark group <package-spec>...`

Marks the specified packages as installed by group. This can be useful if any package was installed as a dependency or a user and is desired to be protected and handled as a group member like during group remove.

Module Command

Command: `module`

Modularity overview is available at man page `dnf.modularity(7)`. Module subcommands take `<module-spec>...` arguments that specify modules or profiles.

`dnf [options] module install <module-spec>...`

Install module profiles, including their packages. In case no profile was provided, all default profiles get installed. Module streams get enabled accordingly.

This command cannot be used for switching module streams. Use the `dnf module switch-to` command for that.

`dnf [options] module update <module-spec>...`

Update packages associated with an active module stream, optionally restricted to a profile. If the `profile_name` is provided, only the packages referenced by that profile will be updated.

`dnf [options] module switch-to <module-spec>...`

Switch to or enable a module stream, change versions of in?

stalled packages to versions provided by the new stream, and re?

move packages from the old stream that are no longer available.

It also updates installed profiles if they are available for the

new stream. When a profile was provided, it installs that pro?

file and does not update any already installed profiles.

This command can be used as a stronger version of the dnf module

enable command, which not only enables modules, but also does a

distrosync to all modular packages in the enabled modules.

It can also be used as a stronger version of the dnf module in?

stall command, but it requires to specify profiles that are sup?

posed to be installed, because switch-to command does not use

default profiles. The switch-to command doesn't only install

profiles, it also makes a distrosync to all modular packages in

the installed module.

`dnf [options] module remove <module-spec>...`

Remove installed module profiles, including packages that were

installed with the dnf module install command. Will not remove

packages required by other installed module profiles or by other

user-installed packages. In case no profile was provided, all

installed profiles get removed.

`dnf [options] module remove --all <module-spec>...`

Remove installed module profiles, including packages that were

installed with the dnf module install command. With --all op?

tion it additionally removes all packages whose names are pro?

vided by specified modules. Packages required by other installed

module profiles and packages whose names are also provided by

any other module are not removed.

`dnf [options] module enable <module-spec>...`

Enable a module stream and make the stream RPMs available in the

package set.

Modular dependencies are resolved, dependencies checked and also

recursively enabled. In case of modular dependency issue the op?

eration will be rejected. To perform the action anyway please

use `--skip-broken` option.

This command cannot be used for switching module streams. Use the `dnf module switch-to` command for that.

`dnf [options] module disable <module-name>...`

Disable a module. All related module streams will become unavailable. Consequently, all installed profiles will be removed and the module RPMs will become unavailable in the package set. In case of modular dependency issue the operation will be rejected. To perform the action anyway please use `--skip-broken` option.

`dnf [options] module reset <module-name>...`

Reset module state so it's no longer enabled or disabled. Consequently, all installed profiles will be removed and only RPMs from the default stream will be available in the package set.

`dnf [options] module provides <package-name-spec>...`

Lists all modular packages matching `<package-name-spec>` from all modules (including disabled), along with the modules and streams they belong to.

`dnf [options] module list [--all] [module_name...]`

Lists all module streams, their profiles and states (enabled, disabled, default).

`dnf [options] module list --enabled [module_name...]`

Lists module streams that are enabled.

`dnf [options] module list --disabled [module_name...]`

Lists module streams that are disabled.

`dnf [options] module list --installed [module_name...]`

List module streams with installed profiles.

`dnf [options] module info <module-spec>...`

Print detailed information about given module stream.

`dnf [options] module info --profile <module-spec>...`

Print detailed information about given module profiles.

`dnf [options] module repoquery <module-spec>...`

List all available packages belonging to selected modules.

`dnf [options] module repoquery --available <module-spec>...`

List all available packages belonging to selected modules.

`dnf [options] module repoquery --installed <module-spec>...`

List all installed packages with same name like packages belong?
ing to selected modules.

Provides Command

Command: `provides`

Aliases: `prov`, `whatprovides`, `wp`

`dnf [options] provides <provide-spec>`

Finds the packages providing the given `<provide-spec>`. This is useful when one knows a filename and wants to find what package (installed or not) provides this file. The `<provide-spec>` is gradually looked for at following locations:

1. The `<provide-spec>` is matched with all file provides of any available package:

```
$ dnf provides /usr/bin/gzip
```

```
gzip-1.9-9.fc29.x86_64 : The GNU data compression program
```

Matched from:

```
Filename : /usr/bin/gzip
```

2. Then all provides of all available packages are searched:

```
$ dnf provides "gzip(x86-64)"
```

```
gzip-1.9-9.fc29.x86_64 : The GNU data compression program
```

Matched from:

```
Provide : gzip(x86-64) = 1.9-9.fc29
```

3. DNF assumes that the `<provide-spec>` is a system command, prepends it with `/usr/bin/`, `/usr/sbin/` prefixes (one at a time) and does the file provides search again. For legacy reasons (packages that didn't do `UsrMove`) also `/bin` and `/sbin` prefixes are being searched:

```
$ dnf provides zless
```

```
gzip-1.9-9.fc29.x86_64 : The GNU data compression program
```

Matched from:

```
Filename : /usr/bin/zless
```

4. If this last step also fails, DNF returns "Error: No Matches found".

This command by default does not force a sync of expired meta? data. See also Metadata Synchronization.

Reinstall Command

Command: `reinstall`

Aliases: `rei`

`dnf [options] reinstall <package-spec>...`

Installs the specified packages, fails if some of the packages are either not installed or not available (i.e. there is no repository where to download the same RPM).

Remove Command

Command: `remove`

Aliases: `rm`

Aliases for explicit NEVRA matching: `remove-n`, `remove-na`, `remove-nevra`

Deprecated aliases: `erase`, `erase-n`, `erase-na`, `erase-nevra`

`dnf [options] remove <package-spec>...`

Removes the specified packages from the system along with any packages depending on the packages being removed. Each <spec> can be either a <package-spec>, which specifies a package directly, or a @<group-spec>, which specifies an (environment) group which contains it. If `clean_requirements_on_remove` is enabled (the default), also removes any dependencies that are no longer needed.

`dnf [options] remove --duplicates`

Removes older versions of duplicate packages. To ensure the integrity of the system it reinstalls the newest package. In some cases the command cannot resolve conflicts. In such cases the `dnf shell` command with `remove --duplicates` and `upgrade dnf-shell` sub-commands could help.

`dnf [options] remove --oldinstallonly`

Removes old installonly packages, keeping only latest versions and version of running kernel.

There are also a few specific remove commands `remove-n`, `remove-na` and `remove-nevra` that allow the specification of an `exact` argument in the NEVRA format.

Remove Examples

```
dnf remove acpi tito
```

Remove the `acpi` and `tito` packages.

```
dnf remove $(dnf repoquery --extras --exclude=tito,acpi)
```

Remove packages not present in any repository, but don't remove the `tito` and `acpi` packages (they still might be removed if they depend on some of the removed packages).

Remove older versions of duplicated packages (an equivalent of yum's `package-cleanup --cleandups`):

```
dnf remove --duplicates
```

Repoinfo Command

Command: `repoinfo`

An alias for the `repolist` command that provides more detailed information like `dnf repolist -v`.

Repolist Command

Command: `repolist`

```
dnf [options] repolist [--enabled|--disabled|--all]
```

Depending on the exact command lists enabled, disabled or all known repositories. Lists all enabled repositories by default.

Provides more detailed information when `-v` option is used.

This command by default does not force a sync of expired metadata. See also [Metadata Synchronization](#).

Repoquery Command

Command: `repoquery`

Aliases: `rq`

Aliases for explicit NEVRA matching: `repoquery-n`, `repoquery-na`, `repoquery-nevra`

```
dnf [options] repoquery [<select-options>] [<query-options>] [<package-file-spec>]
```

Searches available DNF repositories for selected packages and displays the requested information about them. It is an equivalent

lent of rpm -q for remote repositories.

dnf [options] repoquery --groupmember <package-spec>...

List groups that contain <package-spec>.

dnf [options] repoquery --querytags

Provides the list of tags recognized by the --queryformat repo? query option.

There are also a few specific repoquery commands repoquery-n, repoquery-na and repoquery-nevra that allow the specification of an exact argument in the NEVRA format (does not affect arguments of options like --whatprovides <arg>, ...).

Select Options

Together with <package-file-spec>, control what packages are displayed in the output. If <package-file-spec> is given, limits the resulting set of packages to those matching the specification. All packages are considered if no <package-file-spec> is specified.

<package-file-spec>

Package specification in the NEVRA format (name[-[epoch:]ver? sion[-release]][.arch]), a package provide or a file provide.

See Specifying Packages.

-a, --all

Query all packages (for rpmquery compatibility, also a shorthand for repoquery '*' or repoquery without arguments).

--arch <arch>[,<arch>...], --archlist <arch>[,<arch>...]

Limit the resulting set only to packages of selected architectures (default is all architectures). In some cases the result is affected by the basearch of the running system, therefore to run repoquery for an arch incompatible with your system use the --forcearch=<arch> option to change the basearch.

--duplicates

Limit the resulting set to installed duplicate packages (i.e. more package versions for the same name and architecture). In? stallonly packages are excluded from this set.

--unneeded

Limit the resulting set to leaves packages that were installed as dependencies so they are no longer needed. This switch lists packages that are going to be removed after executing the `dnf autoremove` command.

`--available`

Limit the resulting set to available packages only (set by default).

`--disable-modular-filtering`

Disables filtering of modular packages, so that packages of inactive module streams are included in the result.

`--extras`

Limit the resulting set to packages that are not present in any of the available repositories.

`-f <file>`, `--file <file>`

Limit the resulting set only to the package that owns `<file>`.

`--installed`

Limit the resulting set to installed packages only. The `exclude` option in the configuration file might influence the result, but if the command line option `--disableexcludes` is used, it ensures that all installed packages will be listed.

`--installonly`

Limit the resulting set to installed installonly packages.

`--latest-limit <number>`

Limit the resulting set to `<number>` of latest packages for every package name and architecture. If `<number>` is negative, skip `<number>` of latest packages. For a negative `<number>` use the `--latest-limit=<number>` syntax.

`--recent`

Limit the resulting set to packages that were recently edited.

`--repo <repoid>`

Limit the resulting set only to packages from a repository identified by `<repoid>`. Can be used multiple times with accumulative effect.

`--unsatisfied`

Report unsatisfied dependencies among installed packages (i.e. missing requires and existing conflicts).

`--upgrades`

Limit the resulting set to packages that provide an upgrade for some already installed package.

`--userinstalled`

Limit the resulting set to packages installed by the user. The `exclude` option in the configuration file might influence the result, but if the command line option `--disableexcludes` is used, it ensures that all installed packages will be listed.

`--whatdepends <capability>[,<capability>...]`

Limit the resulting set only to packages that require, enhance, recommend, suggest or supplement any of <capabilities>.

`--whatconflicts <capability>[,<capability>...]`

Limit the resulting set only to packages that conflict with any of <capabilities>.

`--whatenhances <capability>[,<capability>...]`

Limit the resulting set only to packages that enhance any of <capabilities>. Use `--whatdepends` if you want to list all dependent packages.

`--whatobsoletes <capability>[,<capability>...]`

Limit the resulting set only to packages that obsolete any of <capabilities>.

`--whatprovides <capability>[,<capability>...]`

Limit the resulting set only to packages that provide any of <capabilities>.

`--whatrecommends <capability>[,<capability>...]`

Limit the resulting set only to packages that recommend any of <capabilities>. Use `--whatdepends` if you want to list all dependent packages.

`--whatrequires <capability>[,<capability>...]`

Limit the resulting set only to packages that require any of

<capabilities>. Use --whatdepends if you want to list all de?
pending packages.

--whatsuggests <capability>[,<capability>...]

Limit the resulting set only to packages that suggest any of
<capabilities>. Use --whatdepends if you want to list all de?
pending packages.

--whatsupplements <capability>[,<capability>...]

Limit the resulting set only to packages that supplement any of
<capabilities>. Use --whatdepends if you want to list all de?
pending packages.

--alldeps

This option is stackable with --whatrequires or --whatdepends
only. Additionally it adds all packages requiring the package
features to the result set (used as default).

--exactdeps

This option is stackable with --whatrequires or --whatdepends
only. Limit the resulting set only to packages that require <ca?
pability> specified by --whatrequires.

--srpm Operate on the corresponding source RPM.

Query Options

Set what information is displayed about each package.

The following are mutually exclusive, i.e. at most one can be speci?
fied. If no query option is given, matching packages are displayed in
the standard NEVRA notation.

-i, --info

Show detailed information about the package.

-l, --list

Show the list of files in the package.

-s, --source

Show the package source RPM name.

--changelogs

Print the package changelogs.

--conflicts

Display capabilities that the package conflicts with. Same as

`--qf "%{conflicts}"`.

`--depends`

Display capabilities that the package depends on, enhances, recommends, suggests or supplements.

`--enhances`

Display capabilities enhanced by the package. Same as `--qf "%{enhances}"`.

`--location`

Show a location where the package could be downloaded from.

`--obsoletes`

Display capabilities that the package obsoletes. Same as `--qf "%{obsoletes}"`.

`--provides`

Display capabilities provided by the package. Same as `--qf "%{provides}"`.

`--recommends`

Display capabilities recommended by the package. Same as `--qf "%{recommends}"`.

`--requires`

Display capabilities that the package depends on. Same as `--qf "%{requires}"`.

`--requires-pre`

Display capabilities that the package depends on for running a %pre script. Same as `--qf "%{requires-pre}"`.

`--suggests`

Display capabilities suggested by the package. Same as `--qf "%{suggests}"`.

`--supplements`

Display capabilities supplemented by the package. Same as `--qf "%{supplements}"`.

`--tree` Display a recursive tree of packages with capabilities specified

by one of the following supplementary options: `--whatrequires`,

--requires, --conflicts, --enhances, --suggests, --provides,
--supplements, --recommends.

--deplist

Produce a list of all direct dependencies and what packages provide those dependencies for the given packages. The result only shows the newest providers (which can be changed by using --verbose).

--nvr Show found packages in the name-version-release format. Same as --qf "%{name}-%{version}-%{release}".

--nevra

Show found packages in the name-epoch:version-release.architecture format. Same as --qf "%{name}-%{epoch}:%{version}-%{release}-%{arch}" (default).

--envra

Show found packages in the epoch:name-version-release.architecture format. Same as --qf "%{epoch}:%{name}-%{version}-%{release}-%{arch}"

--qf <format>, --queryformat <format>

Custom display format. <format> is the string to output for each matched package. Every occurrence of %{<tag>} within is replaced by the corresponding attribute of the package. The list of recognized tags can be displayed by running `dnf repoquery --querytags`.

--recursive

Query packages recursively. Has to be used with --whatrequires <REQ> (optionally with --alldeps, but not with --exactdeps) or with --requires <REQ> --resolve.

--resolve

resolve capabilities to originating package(s).

Examples

Display NEVRAs of all available packages matching light*:

```
dnf repoquery 'light*'
```

Display NEVRAs of all available packages matching name light* and ar?

chitecture noarch (accepts only arguments in the "<name>.<arch>" for?
mat):

```
dnf repoquery-na 'light*.noarch'
```

Display requires of all lighttpd packages:

```
dnf repoquery --requires lighttpd
```

Display packages providing the requires of python packages:

```
dnf repoquery --requires python --resolve
```

Display source rpm of lighttpd package:

```
dnf repoquery --source lighttpd
```

Display package name that owns the given file:

```
dnf repoquery --file /etc/lighttpd/lighttpd.conf
```

Display name, architecture and the containing repository of all
lighttpd packages:

```
dnf repoquery --queryformat '%{name} %{arch} : %{reponame}' lighttpd
```

Display all available packages providing "webserver":

```
dnf repoquery --whatprovides webserver
```

Display all available packages providing "webserver" but only for
"i686" architecture:

```
dnf repoquery --whatprovides webserver --arch i686
```

Display duplicate packages:

```
dnf repoquery --duplicates
```

Display source packages that require a <provide> for a build:

```
dnf repoquery --disablerepo="*" --enablerepo="*-source" --arch=src --whatrequires <provide>
```

Repository-Packages Command

Command: repository-packages

Deprecated aliases: repo-pkgs, repo-packages, repository-pkgs

The repository-packages command allows the user to run commands on top
of all packages in the repository named <reposit>. However, any depen?
dency resolution takes into account packages from all enabled reposito?
ries. The <package-file-spec> and <package-spec> specifications further
limit the candidates to only those packages matching at least one of
them.

The info subcommand lists description and summary information about

packages depending on the packages' relation to the repository. The list subcommand just prints lists of those packages.

```
dnf [options] repository-packages <repoid> check-update [<pack?
age-file-spec>...]
```

Non-interactively checks if updates of the specified packages in the repository are available. DNF exit code will be 100 when there are updates available and a list of the updates will be printed.

```
dnf [options] repository-packages <repoid> info [--all] [<pack?
age-file-spec>...]
```

List all related packages.

```
dnf [options] repository-packages <repoid> info --installed [<pack?
age-file-spec>...]
```

List packages installed from the repository.

```
dnf [options] repository-packages <repoid> info --available [<pack?
age-file-spec>...]
```

List packages available in the repository but not currently installed on the system.

```
dnf [options] repository-packages <repoid> info --extras [<pack?
age-file-specs>...]
```

List packages installed from the repository that are not available in any repository.

```
dnf [options] repository-packages <repoid> info --obsoletes [<pack?
age-file-spec>...]
```

List packages in the repository that obsolete packages installed on the system.

```
dnf [options] repository-packages <repoid> info --recent [<pack?
age-file-spec>...]
```

List packages recently added into the repository.

```
dnf [options] repository-packages <repoid> info --upgrades [<pack?
age-file-spec>...]
```

List packages in the repository that upgrade packages installed on the system.

`dnf [options] repository-packages <repoid> install [<package-spec>...]`

Install packages matching <package-spec> from the repository. If <package-spec> isn't specified at all, install all packages from the repository.

`dnf [options] repository-packages <repoid> list [--all] [<package-spec>...]`

List all related packages.

`dnf [options] repository-packages <repoid> list --installed [<package-spec>...]`

List packages installed from the repository.

`dnf [options] repository-packages <repoid> list --available [<package-spec>...]`

List packages available in the repository but not currently installed on the system.

`dnf [options] repository-packages <repoid> list --extras [<package-spec>...]`

List packages installed from the repository that are not available in any repository.

`dnf [options] repository-packages <repoid> list --obsoletes [<package-spec>...]`

List packages in the repository that obsolete packages installed on the system.

`dnf [options] repository-packages <repoid> list --recent [<package-spec>...]`

List packages recently added into the repository.

`dnf [options] repository-packages <repoid> list --upgrades [<package-spec>...]`

List packages in the repository that upgrade packages installed on the system.

`dnf [options] repository-packages <repoid> move-to [<package-spec>...]`

Reinstall all those packages that are available in the repository.

`dnf [options] repository-packages <repoid> reinstall [<package-spec>...]`

age-spec>...]

Run the `reinstall-old` subcommand. If it fails, run the `move-to` subcommand.

`dnf [options] repository-packages <repoid> reinstall-old [<pack? age-spec>...]`

Reinstall all those packages that were installed from the repository and simultaneously are available in the repository.

`dnf [options] repository-packages <repoid> remove [<package-spec>...]`

Remove all packages installed from the repository along with any packages depending on the packages being removed. If `clean_requirements_on_remove` is enabled (the default) also removes any dependencies that are no longer needed.

`dnf [options] repository-packages <repoid> remove-or-distro-sync [<package-spec>...]`

Select all packages installed from the repository. Upgrade, downgrade or keep those of them that are available in another repository to match the latest version available there and remove the others along with any packages depending on the packages being removed. If `clean_requirements_on_remove` is enabled (the default) also removes any dependencies that are no longer needed.

`dnf [options] repository-packages <repoid> remove-or-reinstall [<package-spec>...]`

Select all packages installed from the repository. Reinstall those of them that are available in another repository and remove the others along with any packages depending on the packages being removed. If `clean_requirements_on_remove` is enabled (the default) also removes any dependencies that are no longer needed.

`dnf [options] repository-packages <repoid> upgrade [<package-spec>...]`

Update all packages to the highest resolvable version available in the repository. When versions are specified in the `<package-spec>`, update to these versions.

`dnf [options] repository-packages <repoid> upgrade-to [<package-specs>...]`

A deprecated alias for the upgrade subcommand.

Search Command

Command: `search`

Aliases: `se`

`dnf [options] search [--all] <keywords>...`

Search package metadata for keywords. Keywords are matched as case-insensitive substrings, globbing is supported. By default lists packages that match all requested keys (AND operation). Keys are searched in package names and summaries. If the "--all" option is used, lists packages that match at least one of the keys (an OR operation). In addition the keys are searched in the package descriptions and URLs. The result is sorted from the most relevant results to the least.

This command by default does not force a sync of expired metadata. See also Metadata Synchronization.

Shell Command

Command: `shell`

Aliases: `sh`

`dnf [options] shell [filename]`

Open an interactive shell for conducting multiple commands during a single execution of DNF. These commands can be issued manually or passed to DNF from a file. The commands are much the same as the normal DNF command line options. There are a few additional commands documented below.

`config [conf-option] [value]`

? Set a configuration option to a requested value. If no value is given it prints the current value.

`repo [list|enable|disable] [repo-id]`

? list: list repositories and their status

? enable: enable repository

? disable: disable repository

transaction [list|reset|solve|run]

? list: resolve and list the content of the transaction

? reset: reset the transaction

? run: resolve and run the transaction

Note that all local packages must be used in the first shell

transaction subcommand (e.g. install /tmp/nodejs-1-1.x86_64.rpm

/tmp/acpi-1-1.noarch.rpm) otherwise an error will occur. Any

disable, enable, and reset module operations (e.g. module enable

nodejs) must also be performed before any other shell transac?

tion subcommand is used.

Swap Command

Command: swap

dnf [options] swap <remove-spec> <install-spec>

Remove spec and install spec in one transaction. Each <spec> can be

either a <package-spec>, which specifies a package directly, or a

@<group-spec>, which specifies an (environment) group which contains

it. Automatic conflict solving is provided in DNF by the --allow?

erasing option that provides the functionality of the swap command

automatically.

Updateinfo Command

Command: updateinfo

Aliases: upif

Deprecated aliases: list-updateinfo, list-security, list-sec, info-updateinfo, info-security, info-sec, summary-updateinfo

dnf [options] updateinfo [--summary|--list|--info] [<availability>]

[<spec>...]

Display information about update advisories.

Depending on the output type, DNF displays just counts of advi?

sory types (omitted or --summary), list of advisories (--list)

or detailed information (--info). The -v option extends the out?

put. When used with --info, the information is even more de?

tailed. When used with --list, an additional column with date of

the last advisory update is added.

<availability> specifies whether advisories about newer versions

of installed packages (omitted or --available), advisories about equal and older versions of installed packages (--installed), advisories about newer versions of those installed packages for which a newer version is available (--updates) or advisories about any versions of installed packages (--all) are taken into account. Most of the time, --available and --updates displays the same output. The outputs differ only in the cases when an advisory refers to a newer version but there is no enabled repository which contains any newer version.

Note, that --available takes only the latest installed versions of packages into account. In case of the kernel packages (when multiple version could be installed simultaneously) also packages of the currently running version of kernel are added.

To print only advisories referencing a CVE or a bugzilla use --with-cve or --with-bz options. When these switches are used also the output of the --list is altered - the ID of the CVE or the bugzilla is printed instead of the one of the advisory.

If given and if neither ID, type (bugfix, enhancement, security/sec) nor a package name of an advisory matches <spec>, the advisory is not taken into account. The matching is case-sensitive and in the case of advisory IDs and package names, globbing is supported.

Output of the --summary option is affected by the autocheck_running_kernel configuration option.

Upgrade Command

Command: upgrade

Aliases: up

Deprecated aliases: update, upgrade-to, update-to, localupdate

`dnf [options] upgrade`

Updates each package to the latest version that is both available and resolvable.

`dnf [options] upgrade <package-spec>...`

Updates each specified package to the latest available version.

Updates dependencies as necessary. When versions are specified in the <package-spec>, update to these versions.

`dnf [options] upgrade @<spec>...`

Alias for the `dnf module update` command.

If the main `obsoletes` configure option is true or the `--obsoletes` flag is present, `dnf` will include package obsoletes in its calculations.

For more information see `obsoletes`.

See also Configuration Files Replacement Policy.

Upgrade-Minimal Command

Command: `upgrade-minimal`

Aliases: `up-min`

Deprecated aliases: `update-minimal`

`dnf [options] upgrade-minimal`

Updates each package to the latest available version that provides a bugfix, enhancement or a fix for a security issue (security).

`dnf [options] upgrade-minimal <package-spec>...`

Updates each specified package to the latest available version that provides a bugfix, enhancement or a fix for security issue (security). Updates dependencies as necessary.

SPECIFYING PACKAGES

Many commands take a <package-spec> parameter that selects a package for the operation. The <package-spec> argument is matched against package NEVRAs, provides and file provides.

<package-file-spec> is similar to <package-spec>, except provides matching is not performed. Therefore, <package-file-spec> is matched only against NEVRAs and file provides.

<package-name-spec> is matched against NEVRAs only.

Globs

Package specification supports the same glob pattern matching that shell does, in all three above mentioned packages it matches against (NEVRAs, provides and file provides).

The following patterns are supported:

- * Matches any number of characters.
- ? Matches any single character.
- [] Matches any one of the enclosed characters. A pair of characters separated by a hyphen denotes a range expression; any character that falls between those two characters, inclusive, is matched. If the first character following the [is a ! or a ^ then any character not enclosed is matched.

Note: Curly brackets ({}) are not supported. You can still use them in shells that support them and let the shell do the expansion, but if quoted or escaped, dnf will not expand them.

NEVRA Matching

When matching against NEVRAs, partial matching is supported. DNF tries to match the spec against the following list of NEVRA forms (in decreasing order of priority):

- ? name-[epoch:]version-release.arch
- ? name.arch
- ? name
- ? name-[epoch:]version-release
- ? name-[epoch:]version

Note that name can in general contain dashes (e.g. package-with-dashes).

The first form that matches any packages is used and the remaining forms are not tried. If none of the forms match any packages, an attempt is made to match the <package-spec> against full package NEVRAs.

This is only relevant if globs are present in the <package-spec>.

<package-spec> matches NEVRAs the same way <package-name-spec> does, but in case matching NEVRAs fails, it attempts to match against provides and file provides of packages as well.

You can specify globs as part of any of the five NEVRA components. You can also specify a glob pattern to match over multiple NEVRA components (in other words, to match across the NEVRA separators). In that case, however, you need to write the spec to match against full package NEVRAs, as it is not possible to split such spec into NEVRA forms.

Specifying NEVRA Matching Explicitly

Some commands (autoremove, install, remove and repoquery) also have aliases with suffixes -n, -na and -nevra that allow to explicitly specify

how to parse the arguments:

? Command install-n only matches against name.

? Command install-na only matches against name.arch.

? Command install-nevra only matches against name-[epoch:]version-release.arch.

SPECIFYING PROVIDES

<provide-spec> in command descriptions means the command operates on packages providing the given spec. This can either be an explicit provide, an implicit provide (i.e. name of the package) or a file provide.

The selection is case-sensitive and globbing is supported.

Specifying File Provides

If a spec starts with either / or */ , it is considered as a potential file provide.

SPECIFYING GROUPS

<group-spec> allows one to select (environment) groups a particular operation should work on. It is a case insensitive string (supporting globbing characters) that is matched against a group's ID, canonical name and name translated into the current LC_MESSAGES locale (if possible).

SPECIFYING MODULES

<module-spec> allows one to select modules or profiles a particular operation should work on.

It is in the form of NAME:STREAM:VERSION:CONTEXT:ARCH/PROFILE and supported partial forms are the following:

? NAME

? NAME:STREAM

? NAME:STREAM:VERSION

? NAME:STREAM:VERSION:CONTEXT

? all above combinations with ::ARCH (e.g. NAME::ARCH)

? NAME:STREAM:VERSION:CONTEXT:ARCH

? all above combinations with /PROFILE (e.g. NAME/PROFILE)

In case stream is not specified, the enabled or the default stream is used, in this order. In case profile is not specified, the system default profile or the 'default' profile is used.

SPECIFYING TRANSACTIONS

<transaction-spec> can be in one of several forms. If it is an integer, it specifies a transaction ID. Specifying last is the same as specifying the ID of the most recent transaction. The last form is last-<offset>, where <offset> is a positive integer. It specifies offset-th transaction preceding the most recent transaction.

PACKAGE FILTERING

Package filtering filters packages out from the available package set, making them invisible to most of dnf commands. They cannot be used in a transaction. Packages can be filtered out by either Exclude Filtering or Modular Filtering.

Exclude Filtering

Exclude Filtering is a mechanism used by a user or by a DNF plugin to modify the set of available packages. Exclude Filtering can be modified by either includepkgs or excludepkgs configuration options in configuration files. The --disableexcludes command line option can be used to override excludes from configuration files. In addition to user-configured excludes, plugins can also extend the set of excluded packages. To disable excludes from a DNF plugin you can use the --disableplugin command line option.

To disable all excludes for e.g. the install command you can use the following combination of command line options:

```
dnf --disableexcludes=all --disableplugin="*" install bash
```

Modular Filtering

Please see the modularity documentation for details on how Modular Filtering works.

With modularity, only RPM packages from active module streams are included in the available package set. RPM packages from inactive module streams, as well as non-modular packages with the same name or provides

as a package from an active module stream, are filtered out. Modular filtering is not applied to packages added from the command line, installed packages, or packages from repositories with `module_hotfixes=true` in their `.repo` file.

Disabling of modular filtering is not recommended, because it could cause the system to get into a broken state. To disable modular filtering for a particular repository, specify `module_hotfixes=true` in the `.repo` file or use `--setopt=<repo_id>.module_hotfixes=true`.

To discover the module which contains an excluded package use `dnf module provides`.

METADATA SYNCHRONIZATION

Correct operation of DNF depends on having access to up-to-date data from all enabled repositories but contacting remote mirrors on every operation considerably slows it down and costs bandwidth for both the client and the repository provider. The `metadata_expire` (see `dnf.conf(5)`) repository configuration option is used by DNF to determine whether a particular local copy of repository data is due to be re-synced. It is crucial that the repository providers set the option well, namely to a value where it is guaranteed that if particular metadata was available in time T on the server, then all packages it references will still be available for download from the server in time $T + \text{metadata_expire}$.

To further reduce the bandwidth load, some of the commands where having up-to-date metadata is not critical (e.g. the `list` command) do not look at whether a repository is expired and whenever any version of it is locally available to the user's account, it will be used. For non-root use, see also the `--cacheonly` switch. Note that in all situations the user can force synchronization of all enabled repositories with the `--refresh` switch.

CONFIGURATION FILES REPLACEMENT POLICY

The updated packages could replace the old modified configuration files with the new ones or keep the older files. Neither of the files are actually replaced. To the conflicting ones RPM gives additional suffix

to the origin name. Which file should maintain the true name after transaction is not controlled by package manager but is specified by each package itself, following packaging guideline.

FILES

Cache Files

`/var/cache/dnf`

Main Configuration

`/etc/dnf/dnf.conf`

Repository

`/etc/yum.repos.d/`

SEE ALSO

? `dnf.conf(5)`, DNF Configuration Reference

? `dnf-PLUGIN(8)` for documentation on DNF plugins.

? `dnf.modularity(7)`, Modularity overview.

? `dnf-transaction-json(5)`, Stored Transaction JSON Format Specification.

? DNF project homepage (<https://github.com/rpm-software-management/dnf/>)

? How to report a bug (<https://github.com/rpm-software-management/dnf/wiki/Bug-Reporting>)

? YUM project homepage (<http://yum.baseurl.org/>)

AUTHOR

See AUTHORS in DNF source distribution.

COPYRIGHT

2012-2023, Red Hat, Licensed under GPLv2+

4.14.0 Mar 15, 2023 DNF(8)