# DNF Documentation

***Release 2.7.3-1***

**See AUTHORS in DNF source distribution.**

**Oct 09, 2017**

# Contents

Contents:

CHAPTER 1

# DNF Use Cases

**Contents**



## Introduction

Every feature present in DNF should be based on a reasonable use case. All the supported use cases are supposed to be enumerated in this document.

In case you use DNF to achieve a goal which is not documented here, either you found an error in the documentation or you misuse DNF. In either case we would appreciate if you share the case with us so we can help you to use DNF in the correct way or add the case to the list. You can only benefit from such a report because then you can be sure that the behavior that you expect will not change without prior notice in the *DNF Release Notes* and that the behavior will be covered by our test suite.

**Important:** Please consult every usage of DNF with our reference documentation to be sure what are you doing. The examples mentioned here are supposed to be as simple as possible and may ignore some minor corner cases.

**Warning:** The list is not complete yet - the use cases are being added incrementally these days.

## General assumptions

The user in question must have the appropriate permissions.

## Ensure that my system contains given mix of features (packages/files/providers/groups)

A system administrator has a list of features that has to be present in an operating system. The features must be provided by RPM packages in system repositories that must be accessible.

A feature may be for example a concrete version of a package (`hawkey-0.5.3-1.fc21.i686`), a pathname of a binary RPM file (`/var/lib/mock/fedora-21-i386/result/hawkey-0.5.3-2.20150116gitd002c90.fc21.i686.rpm`), an URL of a binary RPM file (`http://jenkins.cloud.fedoraproject.org/job/DNF/lastSuccessfulBuild/artifact/fedora-21-i386-build/hawkey-0.5.3-99.649.20150116gitd002c90233fc96893806836a258f14a50ee0cf47.fc21.i686.rpm`), a configuration file (`/etc/yum.repos.d/fedora-rawhide.repo`), a language interpreter (`ruby(runtime_executable)`), an extension (`python3-dnf`), a support for building modules for the current running kernel (`kernel-devel-uname-r = $(uname -r)`), an executable (`*/binaryname`) or a collection of packages specified by any available identifier (`kde-desktop`).

The most recent packages that provide the missing features and suit installation (that are not obsoleted and do not conflict with each other or with the other installed packages) are installed if the given feature is not present already. If any of the packages cannot be installed, the operation fails.

### CLI

```
SPECS="hawkey-0.5.3-1.fc21.i686 @kde-desktop"  # Set the features here.

dnf install $SPECS
```

### Plugins/CLI API

```
"""A plugin that ensures that given features are present."""


import dnf.cli


# The parent class allows registration to the CLI manager.
class Command(dnf.cli.Command):

```

```
    """A command that ensures that given features are present."""

    # An alias is needed to invoke the command from command line.
    aliases = ['foo']  # <-- SET YOUR ALIAS HERE.

    def configure(self):
        """Setup the demands."""
        # Repositories are needed if we want to install anything.
        self.cli.demands.available_repos = True
        # A sack is required by marking methods and dependency resolving.
        self.cli.demands.sack_activation = True
        # Resolving performs a transaction that installs the packages.
        self.cli.demands.resolving = True
        # Based on the system, privileges are required to do an installation.
        self.cli.demands.root_user = True  # <-- SET YOUR FLAG HERE.

    @staticmethod
    def set_argparser(parser):
        """Parse command line arguments."""
        parser.add_argument('package', nargs='+', metavar=_('PACKAGE'),
                            action=OptionParser.ParseSpecGroupFileCallback,
                            help=_('Package to install'))

    def run(self):
        """Run the command."""
        # Feature marking methods set the user request.
        for ftr_spec in self.opts.pkg_specs:
            try:
                self.base.install(ftr_spec)
            except dnf.exceptions.MarkingError:
                raise dnf.exceptions.Error('feature(s) not found: ' + ftr_spec)
        # Package marking methods set the user request.
        try:
            self.base.package_install(self.base.add_remote_rpms(self.opts.filenames,␣
↪strict=False))
        except EnvironmentError as e:
            raise dnf.exceptions.Error(e)
        # Comps data reading initializes the base.comps attribute.
        if self.opts.grp_specs:
            self.base.read_comps(arch_filter=True)
        # Group marking methods set the user request.
        for grp_spec in self.opts.grp_specs:
            group = self.base.comps.group_by_pattern(grp_spec)
            if not group:
                raise dnf.exceptions.Error('group not found: ' + grp_spec)
            self.base.group_install(group, ['mandatory', 'default'])


# Every plugin must be a subclass of dnf.Plugin.
class Plugin(dnf.Plugin):

    """A plugin that registers our custom command."""

    # Every plugin must provide its name.
    name = 'foo'  # <-- SET YOUR NAME HERE.

    # Every plugin must provide its own initialization function.
    def __init__(self, base, cli):
```

```
        """Initialize the plugin."""
        super(Plugin, self).__init__(base, cli)
        if cli:
            cli.register_command(Command)
```

If it makes a sense, the plugin can do the operation in appropriate hooks instead of registering a new command that needs to be called from the command line.

## Extension API

```
"""An extension that ensures that given features are present."""


import sys

import dnf.rpm


if __name__ == '__main__':
    FTR_SPECS = {'hawkey-0.5.3-1.fc21.i686'}  # <-- SET YOUR FEATURES HERE.
    RPM_SPECS = {'./hawkey-0.5.3-1.fc21.i686.rpm'}  # <-- SET YOUR RPMS HERE.
    GRP_SPECS = {'kde-desktop'}  # <-- SET YOUR GROUPS HERE.

    with dnf.Base() as base:
        # Substitutions are needed for correct interpretation of repo files.
        RELEASEVER = dnf.rpm.detect_releasever(base.conf.installroot)
        base.conf.substitutions['releasever'] = RELEASEVER
        # Repositories are needed if we want to install anything.
        base.read_all_repos()
        # A sack is required by marking methods and dependency resolving.
        base.fill_sack()
        # Feature marking methods set the user request.
        for ftr_spec in FTR_SPECS:
            try:
                base.install(ftr_spec)
            except dnf.exceptions.MarkingError:
                sys.exit('Feature(s) cannot be found: ' + ftr_spec)
        # Package marking methods set the user request.
        try:
            base.package_install(base.add_remote_rpms(RPM_SPECS, strict=False))
        except EnvironmentError as e:
            sys.exit(e)
        # Comps data reading initializes the base.comps attribute.
        if GRP_SPECS:
            base.read_comps(arch_filter=True)
        # Group marking methods set the user request.
        for grp_spec in GRP_SPECS:
            group = base.comps.group_by_pattern(grp_spec)
            if not group:
                sys.exit('Group cannot be found: ' + grp_spec)
            base.group_install(group, ['mandatory', 'default'])
        # Resolving finds a transaction that allows the packages installation.
        try:
            base.resolve()
        except dnf.exceptions.DepsolveError:
            sys.exit('Dependencies cannot be resolved.')
```

```
        # The packages to be installed must be downloaded first.
        try:
            base.download_packages(base.transaction.install_set)
        except dnf.exceptions.DownloadError:
            sys.exit('Required package cannot be downloaded.')
        # The request can finally be fulfilled.
        base.do_transaction()
```

## Get a list of available packages filtered by their relation to the system

A system user wants to obtain a list of available RPM packages for their consecutive automatic processing or for informative purpose only. The list of RPM packages is filtered by requested relation to the system or user provided <package-name-specs>. The obtained list of packages is based on available data supplied by accessible system repositories.

A relation to the system might be for example one of the following:

installed - packages already installed on the system

available - packages available in any accessible repository

extras - packages installed on the system not available in any known repository

obsoletes - installed packages that are obsoleted by packages in any accessible repository

recent - packages recently added into accessible repositories

upgrades - available packages upgrading some installed packages

### CLI

```
dnf list *dnf*
dnf list installed *debuginfo
dnf list available gtk*devel
dnf list extras
dnf list obsoletes
dnf list recent
dnf list upgrades
```

### Plugins/CLI API

```
"""A plugin that lists available packages filtered by their relation to the␣
↪system."""

import dnf
import dnf.cli


# If you only plan to create a new dnf subcommand in a plugin
# you can use @dnf.plugin.register_command decorator instead of creating
# a Plugin class which only registers the command
# (for full-fledged Plugin class see examples/install_plugin.py)
@dnf.plugin.register_command
```

```
class Command(dnf.cli.Command):

    """A command that lists packages installed on the system that are
       obsoleted by packages in any known repository."""

    # An alias is needed to invoke the command from command line.
    aliases = ['foo']  # <-- SET YOUR ALIAS HERE.

    def configure(self, args):
        """Setup the demands."""
        # Repositories serve as sources of information about packages.
        self.cli.demands.available_repos = True
        # A sack is needed for querying.
        self.cli.demands.sack_activation = True

    def run(self, args):
        """Run the command."""

        obs_tuples = []
        # A query matches all available packages
        q = self.base.sack.query()

        if not args:
            # Narrow down query to match only installed packages
            inst = q.installed()
            # A dictionary containing list of obsoleted packages
            for new in q.filter(obsoletes=inst):
                obs_reldeps = new.obsoletes
                obsoleted = inst.filter(provides=obs_reldeps).run()
                obs_tuples.extend([(new, old) for old in obsoleted])
        else:
            for pkg_spec in args:
                # A subject serves for parsing package format from user input
                subj = dnf.subject.Subject(pkg_spec)
                # A query restricted to installed packages matching given␣
↪subject
                inst = subj.get_best_query(self.base.sack).installed()
                for new in q.filter(obsoletes=inst):
                    obs_reldeps = new.obsoletes
                    obsoleted = inst.filter(provides=obs_reldeps).run()
                    obs_tuples.extend([(new, old) for old in obsoleted])

        if not obs_tuples:
            raise dnf.exceptions.Error('No matching Packages to list')

        for (new, old) in obs_tuples:
            print('%s.%s obsoletes %s.%s' %
                  (new.name, new.arch, old.name, old.arch))
```

## Extension API

```
"""An extension that lists installed packages not available
   in any remote repository.
"""

import dnf


if __name__ == '__main__':

    with dnf.Base() as base:
        # Repositories serve as sources of information about packages.
        base.read_all_repos()
        # A sack is needed for querying.
        base.fill_sack()

        # A query matches all packages in sack
        q = base.sack.query()

        # Derived query matches only available packages
        q_avail = q.available()
        # Derived query matches only installed packages
        q_inst = q.installed()

        available = q_avail.run()
        for pkg in q_inst.run():
            if pkg not in available:
                print('%s.%s' % (pkg.name, pkg.arch))
```

CHAPTER 2

# 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:

- *autoremove*
- *check*
- *check-update*
- *clean*

- *distro-sync*
- *downgrade*
- *group*
- *help*
- *history*
- *info*
- *install*
- *list*
- *makecache*
- *mark*
- *provides*
- *reinstall*
- *remove*
- *repoinfo*
- *repolist*
- *repoquery*
- *repository-packages*
- *search*
- *shell*
- *swap*
- *updateinfo*
- *upgrade*
- *upgrade-minimal*
- *upgrade-to*

Additional informations:

- *Options*
- *Specifying Packages*
- *Specifying Exact Versions of Packages*
- *Specifying 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>** Includes packages corresponding to the advisory ID, Eg. FEDORA-2201-123. Applicable for install, repoquery, updateinfo, and upgrade command.

**--allowerasing** Allow erasing of installed packages to resolve dependencies. This option could be used as an alternative to `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.

**--bugfix** Includes packages that fix a bugfix issue. Applicable for install, repoquery, updateinfo, and upgrade command.

**--bz=<bugzilla>** Includes packages that fix a Bugzilla ID, Eg. 123123. Applicable for install, repoquery, updateinfo, and upgrade command.

**-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.

**--comment=<comment>** add a comment to transaction history

**-c <config file>, --config=<config file>** config file location

**--cve=<cves>** Includes packages that fix a CVE (Common Vulnerabilities and Exposures) ID (http://cve.mitre.org/about/), Eg. CVE-2201-0123. Applicable for install, repoquery, updateinfo, and upgrade command.

**-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 `./debugdata`.

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

Disable the config file excludes. Takes one of three options:

- `all`, disables all config file excludes
- `main`, disables excludes defined in the `[main]` section
- `repoid`, disables excludes defined for the given repo

**--disableplugin=<plugin names>** Disable the listed plugins specified by names or globs.

**--disablerepo=<repoid>** Disable specific repositories by an id or a glob. This option is mutually exclusive with `--repo`.

**--downloaddir=<path>** Redirect downloaded packages to provided directory. The option has to by used together with *--downloadonly* command line option or with `download` command (dnf-plugins-core).

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

**-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 2. Deprecated, use `-v` instead.

**--enableplugin=<plugin names>** Enable the listed plugins specified by names or globs.

**--enablerepo=<repoid>** Enable additional repositories by an id or a glob.

**--enhancement** Include enhancement relevant packages. Applicable for install, repoquery, updateinfo, and upgrade command.

**-x <package-spec>, --exclude=<package-spec>** Exclude packages specified by `<package-spec>` from the operation.

**--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.

**-h, --help** 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.

- *cachedir*, *log files*, *releasever*, and *gpgkey* are taken from or stored in installroot. *Gpgkeys* are imported into installroot from path, related to the host, described in .repo file.
- *config file* and *reposdir* are searched inside the installroot first. If they are not present, they are taken from host system. Note: When a path is specified within command line argument (`--config=<config file>` in case of *config file* and `--setopt=reposdir=<reposdir>` for *reposdir*) then this path is always related to the host with no exceptions.
- The *pluginpath* and *pluginconfpath* are not related to installroot.

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

  Installroot examples:

  **dnf --installroot=<installroot> --releasever=<release> install system-release**
  Sets permanently the `releasever` of the system within `<installroot>` directory from given `<release>`.

  **dnf --installroot=<installroot> --setopt=reposdir=<path> --config /path/dnf.conf upgra**
  Upgrade packages inside of installroot from repository described by `--setopt` using configuration from `/path/dnf.conf`

**--newpackage** Include newpackage relevant packages. Applicable for install, repoquery, updateinfo, and upgrade command.

**--noautoremove** disable removal of dependencies that are no longer used. It sets *clean_requirements_on_remove* conf option to `False`.

**--nodocs** do not install documentations by using rpm flag 'RPMTRANS_FLAG_NODOCS'

**--nogpgcheck** skip checking GPG signatures on packages

**--noplugins** 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 *obsoletes option*.

Option also affects *repoquery command*, it display's capabilities that the package obsoletes.

Configuration Option: *obsoletes*

**-q, --quiet** In combination with a non-interactive command it shows just the relevant content. It suppresses messages notifying about 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 path or url to a repository (same path as in a baseurl) to add to the repositories for this query. This option can be used multiple times. The repo label for the repository is specified by <repo>. If you want to view only the packages from this repository, combine this with with `--repo=<repo>` or `--disablerepo="*"` switches. The repo label for the repository is specified by <repo>.

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

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

**--sec-severity=<severity>, --secseverity=<severity>** Includes packages that provides a fix for issue of the specified severity. Applicable for install, repoquery, updateinfo, and upgrade command.

**--security** Includes packages that provides a fix for security issue. Applicable for upgrade command.

**--setopt=<option>=<value>** override a config option from the config file. To override config options from repo files, use `repoid.option` for the `<option>`.

**--skip-broken** Resolve depsolve problems by removing packages that are causing problems from the transaction. It is alias for configuration option *strict* with False value.

**--showduplicates** show duplicates, in repos, in list/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>` and `<package-name-spec>` see *Specifying Packages*.

For an explanation of `<package-nevr-spec>` see *Specifying Exact Versions of Packages*.

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

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

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

## Auto Remove Command

`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 *Remove Command* 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 specification of exact argument NEVRA format.

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

## Check Command

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

Checks the local packagedb and produces information on any problems it finds. You can pass the check command the options "–dependencies", "–duplicates", "–obsoleted" or "–provides", to limit the checking that is performed (the default is "all" which does all).

## Check Update Command

`dnf [options] check-update [<package-specs>...]`

Non-interactively checks if updates of the specified packages are available. If no `<package-specs>` are 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.

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` must also ensure the satisfiability of all dependencies and other restrictions.

Output is affected by config option *autocheck_running_kernel*

## Clean Command

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 repo 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.

### Distro-sync command

**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*.

### Distribution-synchronization command

**dnf distribution-synchronization** Deprecated alias for the *Distro-sync command*.

### Downgrade Command

**dnf [options] downgrade <package-installed-specs>...** Downgrades the specified packages to the highest 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.

### Erase Command

**dnf [options] erase <spec>...** Deprecated alias for the *Remove Command*.

### Group Command

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 installed or available from a repo 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 group is already (partially) installed, command installs missing packages from the group.

**dnf [options] group list <group-spec>...** List all matching groups, either among installed or available groups. If nothing is specified list all known groups. Options `--installed` and `--available` narrows down the requested list. Records are ordered by *display_order* tag defined in comps.xml file. Provides a list of all hidden groups by using option `--hidden`. Provides more detailed information when `-v` option is used.

**dnf [options] group remove <group-spec>...** Mark the group removed and remove those packages in the group from the system which are neither comprising 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 pacakges 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 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

**dnf help [<command>]** Displays the help text for all commands. If given a command name then only displays the help for that particular command.

## History Command

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] [<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.

**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-name-spec>** Repeat the specified transaction. Uses the last transaction (with highest ID) if more than one transaction for given <package-name-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 rollback <transaction-spec>|<package-name-spec>** Undo all transactions performed after the specified transaction. Uses the last transaction (with highest ID) if more than one transaction for given <package-name-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 undo <transaction-spec>|<package-name-spec>** Perform the opposite operation to all operations performed in the specified transaction. Uses the last transaction (with highest ID) if more than one transaction for given <package-name-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** It will 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 *Auto Remove Command* or *Remove Command* along with *clean_requirements_on_remove* configuration option set to True is executed. Same results can be accomplished with "dnf repoquery –userinstalled" but repoquery command is much more powerful in formatting of an output.

This command by default does not force a sync of expired metadata. See also *Metadata Synchronization* and *Configuration Files Replacement Policy*.

## Info Command

**dnf [options] info [<package-spec>...]** Is used to list description and summary information about installed and available packages.

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

## Install Command

**dnf [options] install <spec>...** DNF makes sure that the given packages and their dependencies are installed on the system. Each `<spec>` can be either a *<package-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.

When *<package-spec>* that specify 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.

There are also a few specific install commands `install-n`, `install-na` and `install-nevra` that allow to specify exact argument NEVRA format.

See also *Configuration Files Replacement Policy*.

### Install Examples

**dnf install tito** Install package tito (tito is package name).

**dnf install ~/Downloads/tito-0.6.2-1.fc22.noarch.rpm** Install local rpm file tito-0.6.2-1.fc22.noarch.rpm from ~/Downloads/ directory.

**dnf install tito-0.5.6-1.fc22** Install package with specific version. If the package is already installed it will automatically try to downgrade or upgrade to specific version.

**dnf --best install tito** Install the latest available version of package. If the package is already installed it will automatically try to upgrade to the latest version. If the latest version of package cannot be installed, the installation fail.

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

**dnf install https://kojipkgs.fedoraproject.org//packages/tito/0.6.0/1.fc22/noarch/tito-0.6**
Install package directly from URL.

**dnf install '@Web Server'** Install environmental group 'Web Server'

**dnf install /usr/bin/rpmsign** Install a package that provides /usr/bin/rpmsign file.

**dnf -y install tito --setopt=install_weak_deps=False** Install package tito (tito is 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, ...).

## List Command

Dumps 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 it is present in a repository that DNF knows about. The list command can also limit the displayed packages according to other criteria, e.g. to only those that update an

installed package. The *exclude* option in configuration file (.conf) might influence the result, but if the command line option *--disableexcludes* is used, it ensure that all installed packages will be listed.

All the forms take a `[<package-specs>...]` parameter to further limit the result to only those packages matching it.

**dnf [options] list [--all] [<package-name-specs>...]** Lists all packages known to us, present in the RPMDB, in a repo or in both.

**dnf [options] list --installed [<package-name-specs>...]** Lists installed packages.

**dnf [options] list --available [<package-name-specs>...]** Lists available packages.

**dnf [options] list --extras [<package-name-specs>...]** Lists extras, that is packages installed on the system that are not available in any known repository.

**dnf [options] list --obsoletes [<package-name-specs>...]** List the packages installed on the system that are obsoleted by packages in any known repository.

**dnf [options] list --recent [<package-name-specs>...]** List packages recently added into the repositories.

**dnf [options] list --upgrades [<package-name-specs>...]** List upgrades available for the installed packages.

**dnf [options] list --autoremove** List packages which will be removed by `dnf autoremove` command.

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

## Makecache Command

**dnf [options] makecache** Downloads and caches in binary format metadata for all known repos. 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 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

**dnf mark install <package-specs>...** 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 *Auto Remove Command* or *Remove Command* along with *clean_requirements_on_remove* configuration option set to True is executed.

**dnf mark remove <package-specs>...** 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 *Auto Remove 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-specs>...** 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.

## Provides Command

**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.

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

## Reinstall Command

**dnf [options] reinstall <package-specs>...** 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

**dnf [options] remove <package-specs>...** 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 version of duplicated packages.

**dnf [options] remove --oldinstallonly** Removes old installonly packages keeping only installonly_limit latest versions.

There are also a few specific remove commands remove-n, remove-na and remove-nevra that allow to specify exact argument NEVRA format.

### Remove Examples

**dnf remove acpi tito** Remove packages acpi and tito

**dnf remove $(dnf repoquery --extras --exclude=tito,acpi)** Remove packages not present in any repository but it does't remove packages tito and acpi (they still might be removed if they require some of removed packages).

## Repoinfo Command

This command is alias for *repolist* command that provides more detailed information like dnf repolist -v.

## Repolist Command

**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

**dnf [options] repoquery [<select-options>] [<query-options>] [<pkg-spec>]**
Searches the available DNF repositories for selected packages and displays the requested information about them. It is an equivalent of `rpm -q` for remote repositories.

**dnf [options] repoquery --querytags** Provides list of recognized tags by repoquery option *--queryformat*

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

### Select Options

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

**<pkg-spec>** Package specification like: name[-[epoch:]version[-release]][.arch]. See *Specifying Packages*

**-a, --all** Query all packages (for rpmquery compatibility / shorthand for repoquery '*' or repoquery without argument)'

**--arch <arch>[,<arch>...]** Limit the resulting set only to packages of selected architectures.

**--duplicates** Limit the resulting set to installed duplicated packages (i.e. more package versions for the same name and architecture). Installonly 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 `dnf autoremove` command.

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

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

**-f <file>, --file <file>** Limit the resulting set only to package that owns `<file>`.

**--installed** Limit the resulting set to installed packages. The *exclude* option in configuration file (.conf) 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. If negative number is used use syntax `--latest-limit=<number>`.

**--recent** Limit the resulting set to packages that were recently edited.

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

**--unsatisfied** Report unsatisfied dependencies among installed packages (i.e. missing requires and 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 instaled by user. The *exclude* option in configuration file (.conf) might influence the result, but if the command line option *--disableexcludes* is used, it ensures that all installed packages will be listed.

**--whatconflicts <capability>** Limit the resulting set only to packages that conflict `<capability>`.

**--whatenhances <capability>** Limit the resulting set only to packages that enhance <capability>.

**--whatobsoletes <capability>** Limit the resulting set only to packages that obsolete <capability>.

**--whatprovides <capability>** Limit the resulting set only to packages that provide <capability>.

**--whatrecommends <capability>** Limit the resulting set only to packages that recommend <capability>.

**--whatrequires <capability>** Limit the resulting set only to packages that require <capability>.

**--whatsuggests <capability>** Limit the resulting set only to packages that suggest <capability>.

**--whatsupplements <capability>** Limit the resulting set only to packages that supplement <capability>.

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

**--exactdeps** This option is stackable with --whatrequires only. Limit the resulting set only to packages that require <capability> specified by –whatrequires.

**--srpm** Operate on 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 specified. 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 list of files in the package.

**-s, --source** Show package source RPM name.

**--conflicts** Display capabilities that the package conflicts with. Same as --qf "%{conflicts}.

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

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

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

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

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

**--recursive** Query packages recursively. Has to be used with `--whatrequires <REQ>` (optionaly 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 architecture `noarch` (accepts only arguments in format "<name>.<arch>"):

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

Display requires of all ligttpd packages:

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

Display packages providing the requires of python packages:

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

Display source rpm of ligttpd 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 duplicated packages:

```
dnf repoquery --duplicates
```

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

```
dnf remove --duplicates
```

## Repository-Packages Command

The repository-packages command allows the user to run commands on top of all packages in the repository named `<repoid>`. However, any dependency resolution takes into account packages from all enabled repositories. Specifications `<package-name-spec>` and `<package-spec>` further limit the candidates to only those packages matching at least one of them.

`info` subcommand lists description and summary information about packages depending on the packages' relation to the repository. `list` subcommand just dumps lists of that packages.

**dnf [options] repository-packages <repoid> check-update [<package-name-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] [<package-name-spec>...]**
List all related packages.

**dnf [options] repository-packages <repoid> info --installed [<package-name-spec>...]**
List packages installed from the repository.

**dnf [options] repository-packages <repoid> info --available [<package-name-spec>...]**
List packages available in the repository but not currently installed on the system.

**dnf [options] repository-packages <repoid> info --extras [<package-name-specs>...]**
List packages installed from the repository that are not available in any repository.

**dnf [options] repository-packages <repoid> info --obsoletes [<package-name-spec>...]**
List packages in the repository that obsolete packages installed on the system.

**dnf [options] repository-packages <repoid> info --recent [<package-name-spec>...]**
List packages recently added into the repository.

**dnf [options] repository-packages <repoid> info --upgrades [<package-name-spec>...]**
List packages in the repository that upgrade packages installed on the system.

**dnf [options] repository-packages <repoid> install [<package-spec>...]** Install all packages in the repository.

**dnf [options] repository-packages <repoid> list [--all] [<package-name-spec>...]**
List all related packages.

**dnf [options] repository-packages <repoid> list --installed [<package-name-spec>...]**
List packages installed from the repository.

**dnf [options] repository-packages <repoid> list --available [<package-name-spec>...]**
List packages available in the repository but not currently installed on the system.

**dnf [options] repository-packages <repoid> list --extras [<package-name-specs>...]**
List packages installed from the repository that are not available in any repository.

**dnf [options] repository-packages <repoid> list --obsoletes [<package-name-spec>...]**
List packages in the repository that obsolete packages installed on the system.

**dnf [options] repository-packages <repoid> list --recent [<package-name-spec>...]**
List packages recently added into the repository.

**dnf [options] repository-packages <repoid> list --upgrades [<package-name-spec>...]**
List packages in the repository that upgrade packages installed on the system.

**dnf [options] repository-packages <repoid> move-to [<package-name-spec>...]**
Reinstall all those packages that are available in the repository.

**dnf [options] repository-packages <repoid> reinstall [<package-name-spec>...]**
Run `reinstall-old` subcommand. If it fails, run `move-to` subcommand.

**dnf [options] repository-packages <repoid> reinstall-old [<package-name-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-name-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-name-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-name-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-name-spec>...]**
Update all packages to the highest resolvable version available in the repository.

**dnf [options] repository-packages <repoid> upgrade-to <package-nevr-specs>...**
Update packages to the specified versions that are available in the repository. Upgrade-to is deprecated alias for the upgrade subcommand.

## Search Command

**dnf [options] search [--all] <keywords>...** Search package metadata for the keywords. Keywords are matched as case-insensitive substrings, globbing is supported. By default the command will only look at package names and summaries, failing that (or whenever `all` was given as an argument) it will match against 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

**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 config 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

## Swap Command

`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 –allowerasing option that provides functionality of swap command automatically.

## Update Command

**dnf [options] update** Deprecated alias for the *Upgrade Command*.

## Updateinfo Command

**dnf [options] updateinfo [--summary|--list|--info] [<availability>] [<spec>...]**
Display information about update advisories.

Depending on output type, DNF displays just counts of advisory types (omitted or `--summary`), list of advisories (`--list`) or detailed information (`--info`). When `--info` with `-v` option is used, the information is even more detailed.

`<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.

If given and if neither ID, type (`bugfix`, `enhancement`, `security`/`sec`) nor a package name of an advisory does match `<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 option `--summary` is affected by config option *autocheck_running_kernel*

## Upgrade Command

**dnf [options] upgrade** Updates each package to the latest version that is both available and resolvable.

**dnf [options] upgrade <package-installed-specs>...** Updates each specified package to the latest available version. Updates dependencies as necessary.

**dnf [options] upgrade <package-nevr-specs>...** Upgrades packages to the specified versions.

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

**dnf [options] upgrade-minimal** Updates each package to the latest version that provides bugfix, enhancement or fix for security issue (security)

**dnf [options] upgrade-minimal <package-installed-specs>...** Updates each specified package to the latest available version that provides bugfix, enhancement or fix for security issue (security). Updates dependencies as necessary.

### Update-To Command

**dnf [options] update-to <package-nevr-specs>...** Deprecated alias for the *Upgrade Command*.

### Upgrade-To Command

**dnf [options] upgrade-to <package-nevr-specs>...** Deprecated alias for the *Upgrade Command*.

## Specifying Packages

Many commands take a `<package-spec>` parameter that selects a package for the operation. DNF looks for interpretations of the parameter from the most commonly used meanings to the least, that is it tries to see if the given spec fits one of the following patterns (in decreasing order of priority):

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

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

Failing to match the input argument to an existing package name based on the patterns above, DNF tries to see if the argument matches an existing provide.

By default, if multiple versions of the selected package exist in the repo, the most recent version suitable for the given operation is used. If the selected package exists for multiple architectures, the packages which best match the system's architecture will be preferred. The name specification is case-sensitive, globbing characters "`?`, `*` and `[` are allowed and trigger shell-like glob matching. If globbing character is present in `name`, DNF expands given `name` first and consequently selects all packages matching expanded `<package-spec>`.

`<package-name-spec>` is similar to `<package-spec>` except the provides matching is never attempted there.

`<package-installed-specs>` is similar to `<package-specs>` except it considers only installed packages.

## Specifying Exact Versions of Packages

Commands accepting the `<package-nevr-spec>` parameter need not only the name of the package, but also its version, release and optionally the architecture. Further, the version part can be preceded by an epoch when it is relevant (i.e. the epoch is non-zero).

## 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 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 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.

## 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)*) repo config option is used by DNF to determine whether 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 + 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, it will be used. 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)*, assorted DNF plugins that might be installed on the system.
- 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/)

CHAPTER 3

# DNF Configuration Reference

## Description

DNF by default uses the global configuration file at `/etc/dnf/dnf.conf` and all *.repo files found under `/etc/yum.repos.d`. The latter is typically used for repository configuration and takes precedence over global configuration.

The configuration file has INI format consisting of section declaration and `name=value` options below each on separate line. There are two types of sections in the configuration files: main and repository. Main section defines all global configuration options and should be only one.

The repository sections define the configuration for each (remote or local) repository. The section name of the repository in brackets serve as repo ID reference and should be unique across configuration files. The allowed characters of repo ID string are lower and upper case alphabetic letters, digits, `-`, `_`, `.` and `:`. The minimal repository configuration file should aside from repo ID consists of *baseurl*, *metalink* or *mirrorlist* option definition.

## [main] Options

**`assumeyes`** *boolean*

If enabled dnf will assume `Yes` where it would normally prompt for confirmation from user input (see also *defaultyes*). Default is False.

**`autocheck_running_kernel`** *boolean*

Automatic check whether there is installed newer kernel module with security update than currently running kernel. Default is True.

**`best`** *boolean*

When upgrading a package, always try to install its highest version available, even only to find out some of its deps are not satisfiable. Enable this if you want to experience broken dependencies in the repositories firsthand. The default is False.

**check_config_file_age** *boolean*

Specifies whether dnf should automatically expire metadata of repos, which are older than their corresponding configuration file (usually the dnf.conf file and the foo.repo file). Default is `True` (perform the check).

**clean_requirements_on_remove** *boolean*

Remove dependencies that are no longer used during `dnf remove`. A package only qualifies for removal via `clean_requirements_on_remove` if it was installed through DNF but not on explicit user request, i.e. it was pulled in as a dependency. The default is True. (*installonlypkgs* are never automatically removed.)

**config_file_path** *string*

Path to the default main configuration file. Default is `/etc/dnf/dnf.conf`.

**debuglevel** *integer*

Debug messages output level, in the range 0 to 10. The higher the number the more debug output is put to stdout. Default is 2.

**defaultyes** *boolean*

If enabled the default answer to user confirmation prompts will be `Yes`. Not to be confused with *assumeyes* which will not prompt at all. Default is False.

**errorlevel** *integer*

Error messages output level, in the range 0 to 10. The higher the number the more error output is put to stderr. Default is 2. This is deprecated in DNF and overwritten by *--verbose* commandline option.

**exit_on_lock** *boolean*

Should the dnf client exit immediately when something else has the lock. Default is False

**group_package_types** *list*

List of the following: optional, default, mandatory. Tells dnf which type of packages in groups will be installed when 'groupinstall' is called. Default is: default, mandatory

**install_weak_deps** *boolean*

When this option is set to True and a new package is about to be installed, all packages linked by weak dependency relation (Recommends or Supplements flags) with this package will pulled into the transaction. Default is True.

**installonlypkgs** *list*

List of provide names of packages that should only ever be installed, never upgraded. Kernels in particular fall into this category. These packages are never removed by `dnf autoremove` even if they were installed as dependencies (see *clean_requirements_on_remove* for auto removal details). This option append the list values to the default installonlypkgs list used by DNF. The number of kept package versions is regulated by *installonly_limit*.

**installonly_limit** *integer*

Number of *installonly packages* allowed to be installed concurrently. Defaults to 3. The minimal number of installonly packages is 2. Value 0 or 1 means unlimited number of installonly packages.

**keepcache** *boolean*

Keeps downloaded packages in the cache when set to True. Even if it is set to False and packages have not been installed they will still persist until next successful transaction. The default is False.

**logdir** *string*

Directory where the log files will be stored. Default is `/var/log`.

**metadata_timer_sync** time in seconds

The minimal period between two consecutive `makecache timer` runs. The command will stop immediately if it's less than this time period since its last run. Does not affect simple `makecache` run. Use `0` to completely disable automatic metadata synchronizing. The default corresponds to three hours. The value is rounded to the next commenced hour.

**obsoletes** *boolean*

This option only has affect during an install/update. It enables dnf's obsoletes processing logic, which means it makes dnf check whether any dependencies of given package are no longer required and removes them. Useful when doing distribution level upgrades. Default is 'true'.

Command-line option: *–obsoletes*

**pluginconfpath** *list*

List of directories that are searched for plugin configurations to load. All configuration files found in these directories, that are named same as a plugin, are parsed. The default path is `/etc/dnf/plugins`.

**pluginpath** *list*

List of directories that are searched for plugins to load. Plugins found in *any of the directories* in this configuration option are used. The default contains a Python version-specific path.

**protected_packages** *list*

List of packages that DNF should never completely remove. They are protected via Obsoletes as well as user/plugin removals.

The default is: `dnf`, `glob:/etc/yum/protected.d/*.conf` and `glob:/etc/dnf/protected.d/*.conf`. So any packages which should be protected can do so by including a file in `/etc/dnf/protected.d` with their package name in it.

DNF will protect also the package corresponding to the running version of the kernel.

**reposdir** *list*

DNF searches for repository configuration files in the paths specified by `reposdir`. The behavior of `reposdir` could differ when it is used along with *--installroot* option.

**rpmverbosity** *string*

RPM debug scriptlet output level. One of: `critical`, `emergency`, `error`, `warn`, `info` or `debug`. Default is `info`.

**upgrade_group_objects_upgrade** *boolean*

Set this to False to disable the automatic running of `group upgrade` when running the `upgrade` command. Default is True (perform the operation).

## Repo Options

**baseurl** *list*

URLs for the repository.

**cost** *integer*

The relative cost of accessing this repository, defaulting to 1000. This value is compared when the priorities of two repositories are the same. The repository with *the lowest cost* is picked. It is useful to make the library prefer on-disk repositories to remote ones.

**enabled** *boolean*

Include this repository as a package source. The default is True.

**gpgkey** *list* of strings

URLs of a GPG key files that can be used for signing metadata and packages of this repository, empty by default. If a file can not be verified using the already imported keys, import of keys from this option is attempted and the keys are then used for verification.

**metalink** *string*

URL of a metalink for the repository.

**mirrorlist** *string*

URL of a mirrorlist for the repository.

**name** *string*

A human-readable name of the repository. Defaults to the ID of the repository.

**priority** *integer*

The priority value of this repository, default is 99. If there is more than one candidate package for a particular operation, the one from a repo with *the lowest priority value* is picked, possibly despite being less convenient otherwise (e.g. by being a lower version).

**retries** *integer*

Overrides the retries option from the [main] section for this repository.

**skip_if_unavailable** *boolean*

If enabled, DNF will continue running and disable the repository that couldn't be contacted for any reason when downloading metadata. This option doesn't affect skipping of unavailable packages after dependency resolution. To check inaccessibility of repository use it in combination with *refresh command line option*. The default is True.

**strict** *boolean*

If disabled, all unavailable packages or packages with broken dependencies given to DNF command will be skipped without raising the error causing the whole operation to fail. Currently works for install command only. The default is True.

**type** *string*

Type of repository metadata. Supported values are: `rpm-md`. Aliases for `rpm-md`: `rpm`, `repomd`, `rpmmd`, `yum`, `YUM`.

## Repo Variables

Right side of every repo option can be enriched by the following variables:

`$arch`

Refers to the system's CPU architecture e.g, aarch64, i586, i686 and x86_64.

`$basearch`

Refers to the base architecture of the system. For example, i686 and i586 machines both have a base architecture of i386, and AMD64 and Intel64 machines have a base architecture of x86_64.

`$releasever`

Refers to the release version of operating system which DNF derives from information available in RPMDB.

## Options for both [main] and Repo

Some options can be applied in either the main section, per repository, or in a combination. The value provided in the main section is used for all repositories as the default value and concrete repositories can override it in their configuration.

**bandwidth** storage size

Total bandwidth available for downloading. Meaningful when used with the *throttle option*. Storage size is in bytes by default but can be specified with a unit of storage. Valid units are 'k', 'M', 'G'.

**deltarpm** *boolean*

When enabled, DNF will save bandwidth by downloading much smaller delta RPM files, rebuilding them to RPM locally. However, this is quite CPU and I/O intensive. Default is True.

**deltarpm_percentage** *integer*

When the relative size of delta vs pkg is larger than this, delta is not used. Default value is 75 (Deltas must be at least 25% smaller than the pkg). Use *0* to turn off delta rpm processing. Local repositories (with file:// baseurl) have delta rpms turned off by default.

**enablegroups** *boolean*

Determines whether DNF will allow the use of package groups for this repository. Default is True (package groups are allowed).

**excludepkgs** *list*

Exclude packages of this repository, specified by a name or a glob and separated by a comma, from all operations. Can be disabled using `--disableexcludes` command line switch.

**fastestmirror** *boolean*

If enabled a metric is used to find the fastest available mirror. This overrides the order provided by the mirrorlist/metalink file itself. This file is often dynamically generated by the server to provide the best download speeds and enabling fastestmirror overrides this. The default is False.

**gpgcheck** *boolean*

Whether to perform GPG signature check on packages found in this repository. The default is False.

**includepkgs** *list*

Include packages of this repository, specified by a name or a glob and separated by a comma, in all operations. Inverse of *excludepkgs*, DNF will exclude any package in the repository that doesn't match this list. This works in conjunction with exclude and doesn't override it, so if you 'excludepkgs=*.i386' and 'includepkgs=python*' then only packages starting with python that do not have an i386 arch will be seen by DNF in this repo. Can be disabled using `--disableexcludes` command line switch.

**ip_resolve** IP address type

Determines how DNF resolves host names. Set this to '4'/'IPv4' or '6'/'IPv6' to resolve to IPv4 or IPv6 addresses only. By default, DNF resolves to either addresses.

**localpkg_gpgcheck** *boolean*

Whether to perform a GPG signature check on local packages (packages in a file, not in a repositoy). The default is False.

**max_parallel_downloads** *integer*

Maximum number of simultaneous package downloads. Defaults to 3.

**metadata_expire** time in seconds

The period after which the remote repository is checked for metadata update and in the positive case the local metadata cache is updated. The default corresponds to 48 hours. Set this to `-1` or `never` to make the repo never considered expired.

**minrate** storage size

This sets the low speed threshold in bytes per second. If the server is sending data at the same or slower speed than this value for at least *timeout option* seconds, DNF aborts the connection. The default is 1000. Valid units are 'k', 'M', 'G'.

**proxy** *string*

URL of a proxy server to connect through. If none is specified then direct connection is used (the default).

**proxy_username** *string*

The username to use for connecting to the proxy server. Empty by default.

**proxy_password** *string*

The password to use for connecting to the proxy server. Empty by default.

**repo_gpgcheck** *boolean*

Whether to perform GPG signature check on this repository's metadata. The default is False.

**retries** *integer*

Set the number of times any attempt to retrieve a file should retry before returning an error. Setting this to *0* makes dnf try forever. Default is *10*.

**sslcacert** *string*

Path to the directory or file containing the certificate authorities to verify SSL certificates. Empty by default - uses system default.

**sslverify** *boolean*

When enabled, remote SSL connections are verified. If the client can not be authenticated connecting fails and the given repo is not used further. On False, SSL connections can be used but are not verified. Default is True.

**sslclientcert** *string*

Path to the SSL client certificate used to connect to remote sites. Empty by default.

**sslclientkey** *string*

Path to the SSL client key used to connect to remote sites. Empty by default.

**throttle** storage size

Limits the downloading speed. It might be an absolute value or a percentage, relative to the value of the *bandwidth option* option. `0` means no throttling (the default). The absolute value is in bytes by default but can be specified with a unit of storage. Valid units are 'k', 'M', 'G'.

**timeout** time in seconds

Number of seconds to wait for a connection before timing out. Used in combination with *minrate option* option. Defaults to 30 seconds.

**username** *string*

The username to use for connecting to repo with basic HTTP authentication. Empty by default.

**password** *string*

The password to use for connecting to repo with basic HTTP authentication. Empty by default.

## Types of Options

**boolean** This is a data type with only two possible values.

One of following options can be used: 1, 0, True, False, yes, no

**integer** It is a whole number that can be written without a fractional component.

**list** It is an option that could represent one or more strings separated by space or comma characters.

**string** It is a sequence of symbols or digits without any whitespace character.

## Files

**Cache Files** /var/cache/dnf

**Main Configuration File** /etc/dnf/dnf.conf

**Repository** /etc/yum.repos.d/

## See Also

- *dnf(8)*, *DNF Command Reference*

CHAPTER 4

# DNF Automatic

## Synopsis

```
dnf-automatic [<config file>]
```

## Description

Alternative CLI to `dnf upgrade` with specific facilities to make it suitable to be executed automatically and regularly from systemd timers, cron jobs and similar.

The operation of the tool is usually controlled by the configuration file or the function-specific timer units (see below). The command only accepts a single optional argument pointing to the config file, and some control arguments intended for use by the services that back the timer units. If no configuration file is passed from the command line, `/etc/dnf/automatic.conf` is used.

The tool synchronizes package metadata as needed and then checks for updates available for the given system and then either exits, downloads the packages or downloads and applies the packages. The outcome of the operation is then reported by a selected mechanism, for instance via the standard output, email or MOTD messages.

The systemd timer unit `dnf-automatic.timer` will behave as the configuration file specifies (see below) with regard to whether to download and apply updates. Some other timer units are provided which override the configuration file with some standard behaviours:

- dnf-automatic-notifyonly
- dnf-automatic-download
- dnf-automatic-install

Regardless of the configuration file settings, the first will only notify of available updates. The second will download, but not install them. The third will download and install them.

## Run dnf-automatic

You can select one that most closely fits your needs, customize `/etc/dnf/automatic.conf` for any specific behaviors, and enable the timer unit.

For example: `systemctl enable dnf-automatic-notifyonly.timer && systemctl start dnf-automatic-notifyonly.timer`

## Configuration File Format

The configuration file is separated into topical sections.

### `[commands]` section

Setting the mode of operation of the program.

**`apply_updates`** boolean, default: False

> Whether packages comprising the available updates should be applied by `dnf-automatic.timer`, i.e. installed via RPM. Implies `download_updates`. Note that if this is set to `False`, downloaded packages will be left in the cache till the next successful DNF transaction. Note that the other timer units override this setting.

**`download_updates`** boolean, default: False

> Whether packages comprising the available updates should be downloaded by `dnf-automatic.timer`. Note that the other timer units override this setting.

**`upgrade_type`** either one of `default`, `security`, default: `default`

> What kind of upgrades to look at. `default` signals looking for all available updates, `security` only those with an issued security advisory.

**`random_sleep`** time in seconds, default: 300

> Maximal random delay before downloading.

### `[emitters]` section

Choosing how the results should be reported.

**`emit_via`** list, default: `email, stdio, motd`

> List of emitters to report the results through. Available emitters are `stdio` to print the result to standard output, `command` to send the result to a custom command, `command_email` to send an email using a command, and `email` to send the report via email and `motd` sends the result to */etc/motd* file.

**`system_name`** string, default: hostname of the given system

> How the system is called in the reports.

### `[command]` section

The command emitter configuration. Variables usable in format string arguments are `body` with the message body.

**`command_format`** format string, default: `cat`

> The shell command to execute.

**stdin_format** format string, default: `{body}`

The data to pass to the command on stdin.

### [command_email] section

The command email emitter configuration. Variables usable in format string arguments are `body` with message body, `subject` with email subject, `email_from` with the "From:" address and `email_to` with a space-separated list of recipients.

**command_format** format string, default: `mail -s {subject} -r {email_from} {email_to}`

The shell command to execute.

**stdin_format** format string, default: `{body}`

The data to pass to the command on stdin.

**email_from** string, default: `root`

Message's "From:" address.

**email_to** list, default: `root`

List of recipients of the message.

### [email] section

The email emitter configuration.

**email_from** string, default: `root`

Message's "From:" address.

**email_to** list, default: `root`

List of recipients of the message.

**email_host** string, default: `localhost`

Hostname of the SMTP server used to send the message.

### [base] section

Can be used to override settings from DNF's main configuration file. See *DNF Configuration Reference*.

CHAPTER 5

# DNF API Reference

**Contents**



## Introduction

The provided Python API to DNF is supposed to mainly allow writing the following two categories of programs:

1. *plugins* to DNF which extend functionality of the system's DNF installation.
2. extension applications that embed DNF (by importing its Python modules) to perform specific package management tasks.

Please refer to the *DNF Use Cases* where you can find examples of API usage.

**Note:** The API consists of exactly those elements described in this document, items not documented here can change release to release. Opening a bugzilla if certain needed functionality is not exposed is the right thing to do.

## Versioning

DNF follows the Semantic Versioning as defined at http://semver.org/.

This basically means that if your piece of software depends on e.g. DNF 1.1, the requirement can be specified as `1.1 <= dnf < 2`. In other words, you can be sure that your software will be API-compatible with any later release of DNF until the next major version is issued. The same applies for the CLI compatibility. Incompatible API changes are subject to our deprecation policy. Deprecated API items (classes, methods, etc.) are designated as such in the *DNF Release Notes*. The first release where support for such items can be dropped entirely must have, relative to the deprecating release, a higher major version number. DNF will log a warning when a deprecated item is used.

# Contents

API Documentation Contents

## Common Provisions of the DNF API

### Logging

DNF uses the standard Python logging module to do its logging. Three standard loggers are provided:

- `dnf`, used by the core and CLI components of DNF. Messages logged via this logger can end up written to the stdout (console) the DNF process is attached too. For this reason messages logged on the `INFO` level or above should be marked for localization (if the extension uses it).
- `dnf.plugin` should be used by plugins for debugging and similar messages that are generally not written to the standard output streams but logged into the DNF logfile.
- `dnf.rpm` is a logger used by RPM transaction callbacks. Plugins and extensions should not manipulate this logger.

Extensions and plugins can add or remove logging handlers of these loggers at their own discretion.

## `Base`—The centerpiece of DNF

**class** `dnf.Base`

Instances of *`dnf.Base`* are the central point of functionality supplied by DNF. An application will typically create a single instance of this class which it will keep for the runtime needed to accomplish its packaging tasks. Plugins are managed by DNF and get a reference to *`dnf.Base`* object when they run.

*`Base`* instances are stateful objects holding references to various data sources and data sinks. To properly finalize and close off any handles the object may hold, client code should either call *`Base.close()`* when it has finished operations with the instance, or use the instance as a context manager. After the object has left the context, or its *`Base.close()`* has been called explicitly, it must not be used. *`Base.close()`* will delete all downloaded packages upon successful transaction.

**`comps`**

Is `None` by default. Explicit load via *`read_comps()`* initializes this attribute to a *`dnf.comps.Comps`* instance.

**`conf`**

An instance of *`dnf.conf.Conf`*, concentrates all the different configuration options. *`__init__()`* initializes this to usable defaults.

**`repos`**

A *`dnf.repodict.RepoDict`* instance, this member object contains all the repositories available.

**`sack`**

The *`Sack`* that this *`Base`* object is using. It needs to be explicitly initialized by *`fill_sack()`*.

**transaction**
A resolved transaction object, a *dnf.transaction.Transaction* instance, or None if no transaction has been prepared yet.

**__init__**()
Init an instance with a reasonable default configuration. The constructor takes no arguments.

**add_remote_rpms**(*path_list*, *strict=True*)
Add RPM files at list *path_list* to the *sack* and return the list of respective *dnf.package.Package* instances. Does the download to a temporary files for each path if *path* is a remote URL. Raises IOError if there are problems obtaining during reading files and *strict=True*.

**close**()
Close all external handles the object holds. This is called automatically via context manager mechanism if the instance is handled using the with statement.

**init_plugins**([*disabled_glob=None*, *cli=None*])
Initialize plugins. If you want to disable some plugins pass the list of their name patterns to *disabled_glob*. When run from interactive script then also pass your *dnf.cli.Cli* instance.

**pre_configure_plugins**()
Configure plugins by running their pre_configure() method. It makes possible to change variables before repo files and rpmDB are loaded. It also makes possible to create internal repositories that will be affected by --disablerepo and --enablerepo.

**configure_plugins**()
Configure plugins by running their configure() method.

**fill_sack**([*load_system_repo=True*, *load_available_repos=True*])
Setup the package sack. If *load_system_repo* is True, load information about packages in the local RPMDB into the sack. Else no package is considered installed during dependency solving. If *load_available_repos* is True, load information about packages from the available repositories into the sack.

This operation will call *load()* for repos as necessary and can take a long time. Adding repositories or changing repositories' configuration does not affect the information within the sack until *fill_sack()* has been called.

Before this method is invoked, the client application should setup any explicit configuration relevant to the operation. This will often be at least *conf.cachedir* and the substitutions used in repository URLs. See *Conf.substitutions*.

Throws *IOError* exception in case cached metadata could not be opened.

Example:

```
base = dnf.Base()
conf = base.conf
conf.cachedir = CACHEDIR
conf.substitutions['releasever'] = 22
base.repos.add_new_repo('my-repo', conf, baseurl=[MY_REPO_URL])
base.fill_sack()
```

**do_transaction**([*display*])
Perform the resolved transaction. Use the optional *display* object(s) to report the progress. *display* can be either an instance of a subclass of *dnf.callback.TransactionProgress* or a sequence of such instances. Raise *dnf.exceptions.Error* or dnf.exceptions.TransactionCheckError.

**download_packages**(*pkglist*, *progress=None*)
Download packages in *pkglist* from remote repositories. Packages from local repositories or from the command line are not downloaded. *progress*, if given, should be a *DownloadProgress* and can be

used by the caller to monitor the progress of the download. Raises *DownloadError* if some packages failed to download.

**group_install**(*group_id*, *pkg_types*, *exclude=None*)
Mark group with corresponding *group_id* installed and mark the packages in the group for installation. Return the number of packages that the operation has marked for installation. *pkg_types* is a sequence of strings determining the kinds of packages to be installed, where the respective groups can be selected by including `"mandatory"`, `"default"` or `"optional"` in it. If *exclude* is given, it has to be an iterable of package name glob patterns: *group_install()* will then not mark the respective packages for installation whenever possible.

**group_remove**(*group_id*)
Mark group with corresponding *group_id* not installed. All the packages marked as belonging to this group will be marked for removal. Return the number of packages marked for removal in this call.

**group_upgrade**(*group_id*)
Upgrade group with corresponding *group_id*. If there has been packages added to the group's comps information since installing on the system, they will be marked for installation. Similarly, removed packages get marked for removal. The remaining packages in the group are marked for an upgrade. The operation respects the package types from the original installation of the group.

**read_all_repos**()
Read repository configuration from the main configuration file specified by *dnf.conf.Conf.config_file_path* and any `.repo` files under *dnf.conf.Conf.reposdir*. All the repositories found this way are added to *repos*.

**read_comps**(*arch_filter=False*)
Read comps data from all the enabled repositories and initialize the *comps* object. If *arch_filter* is set to `True`, the result is limited to system basearch.

**reset**(***kwargs*)
Reset the state of different *Base* attributes. Selecting attributes to reset is controlled by passing the method keyword arguments set to `True`. When called with no arguments the method has no effect.

| argument passed | effect |
|---|---|
| *goal=True* | drop all the current *packaging requests* |
| *repos=True* | drop the current repositries (see *repos*). This won't affect the package data already loaded into the *sack*. |
| *sack=True* | drop the current sack (see *sack*) |

**resolve**(*allow_erasing=True*)
Resolve the marked requirements and store the resulting *dnf.transaction.Transaction* into *transaction*. Raise *dnf.exceptions.DepsolveError* on a depsolving error. Return `True` if the resolved transaction is non-empty.

Enabling *allow_erasing* lets the solver remove other packages while looking to fulfill the current packaging requests. For instance, this is used to allow the solver to remove dependants of a package being removed.

The exact operation of the solver further depends on the *dnf.conf.Conf.best* setting.

**update_cache**(*timer=False*)
Downloads and caches in binary format metadata for all known repos. Tries to avoid downloading whenever possible (e.g. when the local metadata hasn't expired yet or when the metadata timestamp hasn't changed).

If 'timer' equals 'True', DNF becomes more resource-aware, meaning DNF will not do anything if running on battery power and will terminate immediately if it's too soon after the last successful update_cache operation7.

The *Base* class provides a number of methods to make packaging requests that can later be resolved and turned into a transaction. The *pkg_spec* argument some of them take must be a package specification recognized by *dnf.subject.Subject*. If these methods fail to find suitable packages for the operation they raise a *MarkingError*. Note that successful completion of these methods does not necessarily imply that the desired transaction can be carried out (e.g. for dependency reasons).

**downgrade**(*pkg_spec*)
Mark packages matching *pkg_spec* for downgrade.

**install**(*pkg_spec*)
Mark packages matching *pkg_spec* for installation.

**package_downgrade**(*pkg*, *strict=False*)
If *pkg* is a *dnf.package.Package* in an available repository, mark the matching installed package for downgrade to *pkg*. If strict=False it ignores problems with dep-solving.

**package_install**(*pkg*)
Mark *pkg* (a *dnf.package.Package* instance) for installation. Ignores package that is already installed.

**package_upgrade**(*pkg*)
If *pkg* is a *dnf.package.Package* in an available repository, mark the matching installed package for upgrade to *pkg*.

**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.

**remove**(*pkg_spec*)
Mark packages matching *pkg_spec* for removal.

**upgrade**(*pkg_spec*)
Mark packages matching *pkg_spec* for upgrade.

**upgrade_all**()
Mark all installed packages for an upgrade.

**urlopen(url, repo=None, mode='w+b', **kwargs):**
Open the specified absolute *url* and return a file object which respects proxy setting even for non-repo downloads

## Exceptions

**exception** dnf.exceptions.**Error**

**exception** dnf.exceptions.**DeprecationWarning**
Used to emit deprecation warnings using Python's `warnings.warning()` function.

**exception** dnf.exceptions.**DepsolveError**

**exception** dnf.exceptions.**DownloadError**

**exception** dnf.exceptions.**MarkingError**

**exception** dnf.exceptions.**RepoError**

## Configuration

Configurable settings of the *dnf.Base* object are stored into a *dnf.conf.Conf* instance. The various options are described here.

**class** dnf.conf.**Conf**

**assumeyes**
Boolean option, if set to `True` on any user input asking for confirmation (e.g. after transaction summary) the answer is implicitly `yes`. Default is `False`.

**best**
Boolean option, `True` instructs the solver to either use a package with the highest available version or fail. On `False`, do not fail if the latest version can not be installed. Default is `False`.

**cachedir**
Path to a directory used by various DNF subsystems for storing cache data. Has a reasonable root-writable default depending on the distribution. It is up to the client to set this to a location where files and directories can be created under the running user. The directory can be safely deleted after the *dnf.Base* object is destroyed

**check_config_file_age**
Boolean option. Specifies whether dnf should automatically expire metadata of repos, which are older than their corresponding configuration file (usually the dnf.conf file and the foo.repo file). Default is `True` (perform the check).

**clean_requirements_on_remove**
Boolean option. `True` removes dependencies that are no longer used during `dnf remove`. A package only qualifies for removal via `clean_requirements_on_remove` if it was installed through DNF but not on explicit user request, i.e. it was pulled in as a dependency. The default is `True`. (*installonlypkgs* are never automatically removed.)

**config_file_path**
Path to the default main configuration file. Default is `"/etc/dnf/dnf.conf"`.

**debuglevel**
Debug messages output level, in the range 0 to 10. Default is 2.

**deltarpm_percentage**
Integer option. When the relative size of delta vs pkg is larger than this, delta is not used. Default value is 75 (%). Use *0* to turn off delta rpm processing. Local repositories (with file:// baseurl) have delta rpms always turned off.

**exclude** (*pkgs*)
Exclude packages specified by <pkgs> from the operation.

**exit_on_lock**
Boolean option, if set to `True` dnf client exits immediately when something else has the lock. Default is `False`.

**get_reposdir**
Returns the value of the first valid reposdir or if unavailable the value of created reposdir (string)

**group_package_types**
List of the following: optional, default, mandatory. Tells dnf which type of packages in groups will be installed when 'groupinstall' is called. Default is: default, mandatory

**installonlypkgs**
List of provide names of packages that should only ever be installed, never upgraded. Kernels in particular fall into this category. These packages are never removed by `dnf autoremove` even if they were installed as dependencies (see *clean_requirements_on_remove* for auto removal details). This option overrides the default installonlypkgs list used by DNF. The number of kept package versions is regulated by *installonly_limit*.

**installonly_limit**
An integer to limit the number of installed installonly packages (packages that do not upgrade, instead few versions are installed in parallel). Defaults to `0`, that is the limiting is disabled.

**install_weak_deps**
When this boolean option is set to True and a new package is about to be installed, all packages linked by weak dependency relation (Recommends or Supplements flags) with this package will pulled into the transaction. Default is True.

**installroot**
The root of the filesystem for all packaging operations.

**keepcache**
Keeps downloaded packages in the cache when this boolean option is set to True. Even if it is set to False and packages have not been installed they will still persist until next successful transaction. The default is False.

**logdir**
Directory where the log files will be stored. Default is `"/var/log"`.

**multilib_policy**
Controls how multilib packages are treated during install operations. Can either be `"best"` (the default) for the depsolver to prefer packages which best match the system's architecture, or `"all"` to install all available packages with compatible architectures.

**persistdir**
Directory where the data that DNF keeps track of between different runs is stored. Default is `"/var/lib/dnf"`.

**pluginconfpath**
List of directories that are searched for plugin configuration to load. All configuration files found in these directories, that are named same as a plugin, are parsed. The default contains `/etc/dnf/plugins` path.

**pluginpath**
List of directories where DNF searches for *plugins*. The default contains a Python version-specific path.

**proxy**
URL of of a proxy server to use for network connections. Defaults to `None`, i.e. no proxy used. The expected format of this option is:

```
<scheme>://<ip-or-hostname>[:port]
```

**protected_packages**
List of packages that DNF should never completely remove. They are protected via Obsoletes as well as user/plugin removals.

**proxy_username**
The username to use for connecting to the proxy server. Defaults to `None`.

**proxy_password**
The password to use for connecting to the proxy server. Defaults to `None`.

**releasever**
Used for substitution of `$releasever` in the repository configuration.

**reposdir**
List of directories to search for repo configuration files. Has a reasonable default commonly used on the given distribution.

**retries**
Number of times any attempt to retrieve a file should retry before returning an error. Setting this to *0* makes it try forever. Defaults to *10*.

**sslcacert**
Path to the directory or file containing the certificate authorities to verify SSL certificates. Defaults to None - uses system default.

**sslverify**
Whether SSL certificate checking should be performed at all. Defaults to `True`.

**sslclientcert**
Path to the SSL client certificate used to connect to remote sites. Defaults to None.

**sslclientkey**
Path to the SSL client key used to connect to remote sites. Defaults to None.

**substitutions**
A mapping of substitutions used in repositories' remote URL configuration. The commonly used ones are:

| key | meaning | default |
|---|---|---|
| arch | architecture of the machine | autodetected |
| basearch | the architecture family of the current "arch" | autodetected |
| releasever | release name of the system distribution | `None` |

*`dnf.rpm.detect_releasever()`* can be used to detect the `releasever` value.

Following example shows recommended method how to override autodetected architectures:

```
import dnf
import dnf.arch

base = dnf.Base()
base.conf.substitutions['arch'] = arch
base.conf.substitutions['basearch'] = dnf.rpm.basearch(arch)
base.fill_sack()
...
```

**tsflags**
List of strings adding extra flags for the RPM transaction.

| tsflag | RPM Transaction Flag |
|---|---|
| noscripts | RPMTRANS_FLAG_NOSCRIPTS |
| test | RPMTRANS_FLAG_TEST |
| notriggers | RPMTRANS_FLAG_NOTRIGGERS |
| nodocs | RPMTRANS_FLAG_NODOCS |
| justdb | RPMTRANS_FLAG_JUSTDB |
| nocontexts | RPMTRANS_FLAG_NOCONTEXTS |
| nocrypto | RPMTRANS_FLAG_NOFILEDIGEST |

The `"nocrypto"` option will also set the `_RPMVSF_NOSIGNATURES` and `_RPMVSF_NODIGESTS` VS flags. The `test` option provides a transaction check without performing the transaction. It includes download of packages, gpg keys check (including permanent import of additional keys if necessary), and rpm check to prevent file conflicts.

**username**
The username to use for connecting to repo with basic HTTP authentication. Defaults to `None`.

**upgrade_group_objects_upgrade**
Set this to False to disable the automatic running of `group upgrade` when running the `upgrade` command. Default is `True` (perform the operation).

**password**
The password to use for connecting to repo with basic HTTP authentication. Defaults to `None`.

**basearch**
The base architecture used for installing packages. By default this is auto-detected.

**arch**
The architecture used for installing packages. By default this is auto-detected.

**ignorearch**
If set to `True`, RPM will allow attempts to install packages incompatible with the CPU's architecture. Defaults to `False`.

**prepend_installroot**(*option*)
Prefix config option named *option* with *installroot*.

**read**(*filename=None*)
Read configuration options from the `main` section in *filename*. Option values not present there are left at their current values. If *filename* is `None`, *config_file_path* is used. Conversely, the configuration path used to load the configuration file that was used is stored into *config_file_path* before the function returns.

**dump**()
Print configuration values, including inherited values.

**write_raw_configfile**(*filename*, *section_id*, *substitutions*, *modify*)
Update or create config file. Where *filename* represents name of config file (.conf or .repo); *section_id* represents id of modified section (e.g. main, fedora, updates); *substitutions* represents an instance of base.conf.substitutions; *modify* represents dict of modified options.

## Repository Configuration

**class** dnf.repodict.**RepoDict**
Dictionary mapping repository IDs to the respective *dnf.repo.Repo* objects. Derived from the standard `dict`.

**add**(*repo*)
Add a *Repo* to the repodict.

**all**()
Return a list of all contained repositories.

See the note at *get_matching()* for special semantics of the returned object.

**enable_debug_repos**()
Enable debug repos corresponding to already enabled binary repos.

**enable_source_repos**()
Enable source repos corresponding to already enabled binary repos.

**get_matching**(*key*)
Return a list of repositories which ID matches (possibly globbed) *key* or an empty list if no matching repository is found.

The returned list acts as a composite, transparently forwarding all method calls on itself to the contained repositories. The following thus disables all matching repos:

```
repos = base.repos.get_matching('*-debuginfo')
repos.disable()
```

**iter_enabled**()
Return an iterator over all enabled repos from the dict.

**add_new_repo**(*repoid*, *conf*, *baseurl=()*, ***kwargs*)
Initialize new *Repo* object and add it to the repodict. It requires `repoid` (string), and *dnf.conf.Conf* object. Optionally it can be speciffied baseurl (list), and additionally key/value pairs from *kwargs* to set additional attribute of the *Repo* object. Variables in provided values (`baseurl` or `kwargs`) will be automatically substituted using conf.substitutions (like `$releasever`, ...). It returns the *Repo* object.

`dnf.repo.`**repo_id_invalid**(*repo_id*)
Return index of the first invalid character in the *repo_id* or `None` if all characters are valid. This function is used to validate the section names in `.repo` files.

**class** `dnf.repo.`**Metadata**
Represents the metadata files.

**fresh**
Boolean. `True` if the metadata was loaded from the origin, `False` if it was loaded from the cache.

**class** `dnf.repo.`**Repo**
Repository object used for metadata download. To configure it properly one has to give it either *metalink*, *mirrorlist* or *baseurl* parameter.

**baseurl**
List of URLs for this repository. Defaults to `[]`.

**cost**
The relative cost of accessing this repository, defaulting to 1000. This value is compared when the priorities of two repositories are the same. The repository with *the lowest cost* is picked. It is useful to make the library prefer on-disk repositories to remote ones.

**excludepkgs**
List of packages specified by a name or a glob. DNF will exclude every package in the repository that does match this list from all operations. Defaults to `[]`.

**id**
ID of this repo.

**includepkgs**
List of packages specified by a name or a glob. DNF will exclude any package in the repository that doesn't match this list. This works in conjunction with exclude and doesn't override it, so if you 'excludepkgs=*.i386' and 'includepkgs=python*' then only packages starting with python that do not have an i386 arch will be seen by DNF in this repo. Defaults to `[]`.

**metadata**
If *load()* has been called and succeeded, this contains the relevant *Metadata* instance.

**metalink**
URL of a metalink for this repository. Defaults to `None`

**mirrorlist**
URL of a mirrorlist for this repository. Defaults to `None`

**name**
A string with the repo's name. By default it has value of repo's ID.

**pkgdir**
Directory where packages of a remote repo will be downloaded to. By default it is derived from *cachedir* in *__init__()* but can be overridden by assigning to this attribute.

**proxy**
URL of of a proxy server to use when connecting to this repo. Defaults to `None`, i.e. no proxy used. Also see *Conf.proxy*.

**proxy_username**
The username to use for connecting to the proxy server. Defaults to `None`.

**proxy_password**
The password to use for connecting to the proxy server. Defaults to `None`.

**repofile**
The path to configuration file of the class.

**skip_if_unavailable**
If enabled, DNF will continue running and disable the repository that couldn't be contacted for any reason when downloading metadata. This option doesn't affect skipping of unavailable packages after dependency resolution. The default is `True`.

**sslcacert**
Path to the directory or file containing the certificate authorities to verify SSL certificates. Defaults to None - uses system default.

**sslverify**
Whether SSL certificate checking should be performed at all. Defaults to `True`.

**sslclientcert**
Path to the SSL client certificate used to connect to remote sites. Defaults to None.

**sslclientkey**
Path to the SSL client key used to connect to remote sites. Defaults to None.

**username**
The username to use for connecting to repo with basic HTTP authentication. Defaults to `None`.

**password**
The password to use for connecting to repo with basic HTTP authentication. Defaults to `None`.

**__init__**(*name*, *parent_conf*)
Init repository with ID *name* and the *parent_conf* which an instance of *dnf.conf.Conf* holding main dnf configuration.

**disable**()
Disable the repository. Repositories are enabled by default.

**dump**()
Print repository configuration, including inherited values.

**enable**()
Enable the repository (the default).

**load**()
Load the metadata of this repository. Will try to use local cache if possible and initiate and finish download if not. Returns `True` if fresh metadata has been downloaded and `False` if cache was used. Raises *dnf.exceptions.RepoError* if the repo metadata could not be obtained.

**set_progress_bar**(*progress*)
Set the download progress reporting object for this repo during *load()*. *progress* must be an instance of *dnf.callback.DownloadProgress*.

## Sack

**class** dnf.sack.**Sack**
The package sack. Contains metadata information about all known packages, installed and available.

**query**()
Return a *Query* for querying packages contained in this sack.

## Queries and Subjects

**class** dnf.query.**Query**
Facilitates lookup of packages in a *Sack* based on given criteria. Query actually does not consult the information in the Sack until it is evaluated. The evaluation happens either explicitly using *run()* or by iterating the query, for example:

```
q = base.sack.query()
i = q.installed()
i = i.filter(name='pepper')
packages = list(i) # i only gets evaluated here

a = q.available()
a = a.filter(name='pepper')
for pkg in a: # a only gets evaluated here
    print(pkg.name)
```

Notice that none of the filtering methods mutates the state of the *Query* but produces a new object instead.

**available**()
Return a new query limiting the original query to the not-installed packages, that is packages available from the repositories.

**downgrades**()
Return a new query that limits the result only to packages that can be downgrade candidates to other packages in the current set. Downgrade candidate has the same name, lower EVR and the architecture of the original and the downgrade candidate are suitable for a downgrade. Specifically, the filtering does not take any steps to establish that the downgrade candidate can actually be installed.

**duplicated**()
Return a new query that limits the result only to installed packages of same name and different version. Optional argument exclude accepts a list of package names that will be excluded from result.

**extras**()
Return a new query that limits the result to installed packages that are not present in any repo

**filter**(***kwargs*)
Return a new query limiting the original query to the key/value pairs from *kwargs*. Multiple *kwargs* can be passed, the filter then works by applying all of them together (logical AND). Values inside of list or query are cumulative (logical OR).

Allowed keys are:

| key | value type | value meaning |
|---|---|---|
| arch | string | match against packages' architecture |
| downgrades | boolean | see *downgrades()*. Defaults to `False`. |
| empty | boolean | `True` limits to empty result set. Defaults to `False`. |
| epoch | integer | match against packages' epoch. |
| file | string | match against packages' files |
| latest | boolean | see *latest()*. Defaults to `False`. |
| name | string | match against packages' names |
| release | string | match against packages' releases |
| reponame | string | match against packages repositories' names |
| version | string | match against packages' versions |
| obsoletes | Query | match packages that obsolete any package from query |
| pkg | Query | match against packages in query |
| pkg* | list | match against hawkey.Packages in list |
| provides | string | match against packages' provides |
| provides* | Hawkey.Reldep | match against packages' provides |
| requires | string | match against packages' requirements |
| requires* | Hawkey.Reldep | match against packages' requirements |
| upgrades | boolean | see *upgrades()*. Defaults to `False`. |

*The key can also accept a list of values with specified type.

The key name can be supplemented with a relation-specifying suffix, separated by __:

| key suffix | value type | semantics |
|---|---|---|
| eq | any | exact match; This is the default if no suffix is specified. |
| glob | string | shell-style wildcard match |
| gt | integer | the actual value is greater than specified |
| gte | integer | the actual value is greater than or equal to specified |
| lt | integer | the actual value is less than specified |
| lte | integer | the actual value is less than or equal to specified |
| neq | any | does not equal |
| substr | string | the specified value is contained in the actual value |

For example, the following creates a query that matches all packages containing the string "club" in its name:

```
q = base.sack.query().filter(name__substr="club")
```

**installed**()
Return a new query that limits the result to the installed packages only.

**latest**(*limit=1*)
Return a new query that limits the result to `limit` highest version of packages per package name and per architecture.

**run**()
Evaluate the query. Returns a list of matching *dnf.package.Package* instances.

**upgrades**()
Return a new query that limits the result only to packages that can be upgrade candidates to at least one package in the current set. Upgrade candidate has the same name, higher EVR and the architectures of the original and the upgrade candidate package are suitable for an upgrade. Specifically, the filtering does not take any steps to establish that the upgrade candidate can actually be installed.

**class** dnf.subject.**Subject**
As *explained on the DNF man page*, users of the CLI are able to select packages for an operation in different

formats, leaving seemingly arbitrary parts out of the spec and even using globbing characters. This class implements a common approach to parsing such input and produce a *Query* listing all packages matching the input or a *Selector* selecting a single package that best matches the input given a transaction operation.

**__init__** (*pkg_spec*, *ignore_case=False*)
Initialize the *Subject* with *pkg_spec* input string with following *semantic*. If *ignore_case* is `True` ignore the case of characters in *pkg_spec*.

**get_best_query** (*sack*, *with_nevra=True*, *with_provides=True*, *with_filenames=True*, *forms=None*)
Return a `Query` yielding packages matching the given input. The result of the returned query can be an empty set if no package matches. *sack* is the *Sack* that the returned query will search. *with_nevra* enable search by nevra, *with_provides* indicates whether besides package names also packages' provides are searched for a match, and *with_filenames* indicates whether besides package provides also packages' file provides are searched for a match. *forms* is a list of pattern forms from hawkey. Leaving the parameter to `None` results in using a reasonable default list of forms.

**get_best_selector** (*sack*, *forms=None*, *obsoletes=True*, *reponame=None*, *reports=False*)
Return a *Selector* that will select a single best-matching package when used in a transaction operation. *sack* and *forms* have the same meaning as in *get_best_query()*. If `obsoletes`, selector will also contain packages that obsoletes requested packages (default is True). If `reponame`, the selection of available packages is limited to packages from that repo (default is False). If `reports`, it will report if packages where already installed (default is False).

**get_nevra_possibilities** (*self*, *forms=None*)
Return generator for every possible nevra. Each possible nevra is represented by NEVRA class (libdnf) that has attributes name, epoch, version, release, arch. *forms* have the same meaning as in *get_best_query()*.

Example how to use it when it is known that string could be full NEVRA or NEVR:

```
subject = dnf.subjet.Subject("my_nevra_string")
possible_nevra = subject.get_nevra_possibilities(forms=[hawkey.FORM_NEVRA,␣
↪hawkey.FORM_NEVR])
```

To print all possible names use:

```
for nevra in possible_nevra:
    print(nevra.name)
```

## Selector

**class** dnf.selector.**Selector**
Specify a target of a transaction operation.

## Package

**class** dnf.package.**Package**
Represents a unit of software management, typically corresponds to an RPM file.

**arch**
Architecture of the package (string).

**buildtime**
Seconds since the epoch when the package was built (integer).

**debug_name**
The name of the gebug-info package (string).

**downloadsize**
The size of rpm package in bytes (integer).

**epoch**
Epoch of the package (integer)

**files**
Files the package provides (list of strings)

**installtime**
Seconds since the epoch when the package was installed (integer).

**installsize**
Space in bytes the package takes on the system after installation (integer).

**remote_location** (*schemes=('http', 'ftp', 'file', 'https')*)
The location from where the package can be downloaded from (string). If information unavailable it returns `None`. `schemes` limits result to list of protocols.

**name**
The name of the package (string).

**obsoletes**
Packages that are obsoleted by the package (list of Hawkey.Reldep).

**provides**
Package's provides (list of Hawkey.Reldep).

**release**
Release of the package (string).

**requires**
Package's requirements (list of Hawkey.Reldep).

**source_debug_name**
The name of the source gebug-info package (string).

**source_name**
The name of the source package (string).

**sourcerpm**
Full name of the SRPM used to build this package (string).

**version**
Version of the package (string).

## Transaction

**class** dnf.transaction.**TransactionItem**

**installs**()
Return *packages* that will get added onto the system by this transaction item.

**removes**()
Return *packages* that will get removed from the system by this transaction item.

**class** dnf.transaction.**Transaction**
Instances of this class describe a resolved transaction set. The transaction object can be iterated for the contained *items*.

The packaging requests from the contained items are later passed to the core package manager (RPM) as they are without further dependency resolving. If the set is not fit for an actual transaction (e.g. introduces conflicts, has inconsistent dependencies) RPM then by default refuses to proceed.

**install_set**
Read-only property which contains set of *Packages* to be installed.

**remove_set**
Read-only property which contains set of *Packages* to be removed.

**add_downgrade**(*new*, *downgraded*, *obsoleted*)
Add a downgrade operation to the transaction. *new* is a *Package* to downgrade to, *downgraded* is the installed *Package* being downgraded, *obsoleted* is a list of installed *Packages* that are obsoleted by the *downgrade* (or `None` for no obsoletes).

**add_erase**(*erased*)
Add an erase operation to the transaction. *erased* is a *Package* to erase.

**add_install**(*new*, *obsoleted*, *reason='unknown'*)
Add an install operation to the transaction. *new* is a *Package* to install, *obsoleted* is a list of installed *Packages* that are obsoleted by *new* (or `None` for no obsoletes). *reason*, if provided, must be either `'dep'` for a package installed as a dependency, `'user'` for a package installed per user's explicit request or `'unknown'` for cases where the package's origin can not be decided. This information is stored in the DNF package database and used for instance by the functionality that removes excess packages (see *clean_requirements_on_remove*).

**add_reinstall**(*new*, *reinstalled*, *obsoleted*)
Add a reinstall operation to the transaction. *new* is a *Package* to reinstall over the installed *reinstalled*. *obsoleted* is a list of installed *Packages* that are obsoleted by *new*.

**add_upgrade**(*upgrade*, *upgraded*, *obsoleted*)
Add an upgrade operation to the transaction. *upgrade* is a *Package* to upgrade to, *upgraded* is the installed *Package* to be upgraded, *obsoleted* is a list of installed *Packages* that are obsoleted by the *upgrade*.

## Comps, or the Distribution Compose Metadata

**class** dnf.comps.**Comps**
An object of this class can merge comps information from arbitrary repositories. It typically is instantiated from *dnf.Base* and covers all the available repositories.

The `*_by_pattern` methods all take a *pattern* and an optional *case_sensitive* parameter. The pattern is matched against names and IDs of objects in the domain (groups, categories, environments), the globbing characters in *pattern* retain their usual expanding meaning. If *case_sensitive* is `True`, matching is done in a case-sensitive manner.

**categories**
List of all contained *dnf.comps.Category* objects.

**environments**
List of all contained *dnf.comps.Environment* objects ordered by *display_order* tag defined in comps.xml file.

**groups**
List of all contained *dnf.comps.Group* objects ordered by *display_order* tag defined in comps.xml file.

**category_by_pattern**(*pattern*, *case_sensitive=False*)
Returns a *dnf.comps.Category* object matching *pattern*, or `None`.

**categories_by_pattern**(*pattern*, *case_sensitive=False*)
Return an iterable of *dnf.comps.Category* objects matching *pattern*.

**categories_iter**()
Return iterator over all contained *dnf.comps.Category* objects.

**environment_by_pattern**(*pattern*, *case_sensitive=False*)
Return a *dnf.comps.Environment* object matching *pattern*, or `None`.

**environments_by_pattern**(*pattern*, *case_sensitive=False*)
Return an iterable of *dnf.comps.Environment* objects matching *pattern* ordered by *display_order* tag defined in comps.xml file.

**environments_iter**
Return iterator over all contained *dnf.comps.Environment* objects in order they appear in comps.xml file.

**group_by_pattern**(*pattern*, *case_sensitive=False*)
Return a *dnf.comps.Group* object matching *pattern*, or `None`.

**groups_by_pattern**(*pattern*, *case_sensitive=False*)
Return an iterable of *dnf.comps.Group* objects matching *pattern* ordered by *display_order* tag defined in comps.xml file.

**groups_iter**
Return iterator over all contained *dnf.comps.Group* objects in order they appear in comps.xml file.

**class** dnf.comps.**Package**
Represents comps package data.

---

**Note:** Should not be confused with *dnf.package.Package* which represents a package contained in a *Sack*. There is no guarantee whether the comps package has a corresponding real sack package, i.e. there can be no package of given name in the sack, one such package, or more than one. For this reason two separate types are introduced.

---

**name**
Name of the package.

**option_type**
The type of inclusion of this particular package in its group. Must be one of the *inclusion types*.

**class** dnf.comps.**Category**

**id**
Unique identifier of the category.

**name**
Name of the category.

**ui_name**
The name of the category translated to the language given by the current locale.

**ui_description**
The description of the category translated to the language given by the current locale.

**class** dnf.comps.**Environment**
Has the same set of attributes as *dnf.comps.Category*.

**class** dnf.comps.**Group**
Has the same set of attributes as *dnf.comps.Category*.

**packages_iter**()
Return iterator over all *packages* belonging in this group.

Following types of inclusions of objects in their parent objects are defined:

dnf.comps.**CONDITIONAL**

dnf.comps.**DEFAULT**

dnf.comps.**MANDATORY**

dnf.comps.**OPTIONAL**

## Plugin Interface

DNF plugin can be any Python class fulfilling the following criteria:

1. it derives from *dnf.Plugin*,
2. it is made available in a Python module stored in one of the *Conf.pluginpath*,
3. provides its own *name* and *__init__()*.

When DNF CLI runs it loads the plugins found in the paths during the CLI's initialization.

**class** dnf.**Plugin**
The base class all DNF plugins must derive from.

**name**
Plugin must set this class variable to a string identifying the plugin. The string can only contain alphanumeric characters and underscores.

**static read_config**(*conf*)
Read plugin's configuration into a ConfigParser compatible instance. *conf* is a *Conf* instance used to look up the plugin configuration directory.

**__init__**(*base*, *cli*)
Plugin must override this. Called immediately after all the plugins are loaded. *base* is an instance of *dnf.Base*. *cli* is an instance of *dnf.cli.Cli* but can also be `None` in case DNF is running without a CLI (e.g. from an extension).

**config**()
Plugin can override this. This hook is called immediately after the CLI/extension is finished configuring DNF. The plugin can use this to tweak the global configuration or the repository configuration.

**resolved**()
Plugin can override this. This hook is called immediately after the CLI has finished resolving a transaction. The plugin can use this to inspect the resolved but not yet executed `Base.transaction`.

**sack**()
Plugin can override this. This hook is called immediately after *Base.sack* is initialized with data from all the enabled repos.

**pre_transaction**()
Plugin can override this. This hook is called just before transaction execution. This means after a successful transaction test. RPMDB is locked during that time.

**transaction**()
Plugin can override this. This hook is called immediately after a successful transaction.

You may want to see the comparison with yum plugin hook API.

## Progress Reporting with Callbacks

**class** dnf.callback.**Payload**
Represents one item (file) from the download batch.

**__str__**()
Provide concise, human-readable representation of this Payload.

**download_size**()
Total size of this Payload when transferred (e.g. over network).

**class** dnf.callback.**DownloadProgress**
Base class providing callbacks to receive information about an ongoing download.

**end**(*payload*, *status*, *msg*)
Report finished download of a *payload*, `Payload` instance. *status* is a constant with the following meaning:

| *status* value | meaning |
|---|---|
| STATUS_OK | Download finished successfully. |
| STATUS_DRPM | DRPM rebuilt successfully. |
| STATUS_ALREADY_EXISTS | Download skipped because the local file already exists. |
| STATUS_MIRROR | Download failed on the current mirror, will try to use next mirror in the list. |
| STATUS_FAILED | Download failed because of another error. |

*msg* is an optional string error message further explaining the *status*.

**progress**(*payload*, *done*)
Report ongoing progress on the given *payload*. *done* is the number of bytes already downloaded from *payload*.

**start**(*total_files*, *total_size*, *total_drpms=0*)
Report start of a download batch. *total_files* is the total number of payloads in the batch. *total_size* is the total number of bytes to be downloaded. *total_drpms* is the total number of drpms payloads in the batch.

**class** dnf.callback.**TransactionProgress**
Base class providing callbacks to receive information about an ongoing transaction.

**error**(*message*)
Report an error that occurred during the transaction. *message* is a string which describes the error.

**progress**(*package*, *action*, *ti_done*, *ti_total*, *ts_done*, *ts_total*)
Report ongoing progress on the given transaction item. *package* is the `dnf.package.Package` being processed and *action* is a constant with the following meaning:

| *action* value | meaning | Appear-ance* |
|---|---|---|
| PKG_CLEANUP | *package* cleanup is being performed. | 3 |
| PKG_DOWNGRADE | *package* is being downgraded. | 2 |
| PKG_INSTALL | *package* is being installed. | 2 |
| PKG_OBSOLETE | *package* is being obsoleted. | 2 |
| PKG_REINSTALL | *package* is being reinstalled. | 2 |
| PKG_REMOVE | *package* is being removed. | 2 |
| PKG_UPGRADE | *package* is being upgraded. | 2 |
| PKG_VERIFY | *package* is being verified. | 5 |
| PKG_SCRIPTLET | *package* scriplet is being performed. | Anytime |
| TRANS_PREPARATION | *transaction* is being prepared. | 1 |
| TRANS_POST | The post-trans phase started. In this case, all the other arguments are `None`. | 4 |

*This is order in which state of transaction which callback action can appear. Only PKG_SCRIPTLET can appear anytime during transaction even before transaction starts.

*ti_done* is the number of processed bytes of the transaction item, *ti_total* is the total number of bytes of the transaction item, *ts_done* is the number of actions processed in the whole transaction and *ts_total* is the total number of actions in the whole transaction.

## RPM Interface

`dnf.rpm.`**`detect_releasever`**`(`*installroot*`)`
Return the release name of the distribution of the tree rooted at *installroot*. The function uses information from RPMDB found under the tree.

Returns `None` if the information can not be determined (perhaps because the tree has no RPMDB).

`dnf.rpm.`**`basearch`**`(`*arch*`)`
Return base architecture of the processor based on *arch* type given. E.g. when *arch* i686 is given then the returned value will be i386.

## Command Line Interface Hooks

*`dnf.cli`* is a part of DNF that contains code handling the command line tasks for DNF, like for instance `dnf install emacs`, and outputs the results to the terminal. It is usually of no interest for DNF extension applications, but some parts of it described here can be used by the *Plugin Interface* to hook up custom commands.

When packaging your custom command, we recommend you to define a virtual provide in the form of `Provides: dnf-command(<alias>)` in the spec file. See *the virtual provides usage* for the details.

**exception** `dnf.cli.`**`CliError`**
Signals a CLI-specific problem (reading configuration, parsing user input, etc.). Derives from *`dnf.exceptions.Error`*.

**class** `dnf.cli.demand.`**`DemandSheet`**
Instances are used to track requests of commands and plugins about how CLI should set up/handle other parts of CLI processing that are not under the command's/plugin's direct control. The boolean attributes of the sheet can not be reset once explicitly set, doing so raises an `AttributeError`.

**`allow_erasing`**
If `True`, the dependnecy solver is allowed to look for solutions that include removing other packages while looking to fulfill the current packaging requests. Defaults to `False`. Also see *`dnf.Base.resolve()`*.

**available_repos**
If `True` during sack creation (*sack_activation*), download and load into the sack the available repositories. Defaults to `False`.

**resolving**
If `True` at a place where the CLI would otherwise successfully exit, resolve the transaction for any outstanding packaging requests before exiting. Defaults to `False`.

**root_user**
`True` informs the CLI that the command can only succeed if the process's effective user id is `0`, i.e. root. Defaults to `False`.

**sack_activation**
If `True`, demand that the CLI sets up the *Sack* before the command's *run()* method is executed. Defaults to `False`.

Depending on other demands and the user's configuration, this might or might not correctly trigger metadata download for the available repositories.

**success_exit_status**
The return status of the DNF command on success. Defaults to `0`.

**transaction_display**
An additional instance of a subclass of *dnf.callback.TransactionProgress* used to report information about an ongoing transaction.

**class** dnf.cli.**Command**
Base class of every DNF command.

**aliases**
Sequence of strings naming the command from the command line. Must be a class variable. The list has to contain at least one string, the first string in the list is considered the canonical name. A command name can be contain only letters and dashes providing the name doesn't start with a dash.

**base**
The *dnf.Base* instance to use with this command.

**cli**
The *dnf.cli.Cli* instance to use with this command.

**summary**
One line summary for the command as displayed by the CLI help.

**__init__**(*cli*)
Command constructor which can be overridden. The constructor is called during CLI configure phase when one of the command's aliases is parsed from *dnf* commandline. *cli* is an instance of *dnf.cli.Cli*.

**pre_configure**()
Perform any pre-configuration on the command itself and on the CLI. Typically, the command implements this call to set up releasever or enable/disable repository. This method is called before configuration of repos.

**configure**()
Perform any configuration on the command itself and on the CLI. Typically, the command implements this call to set up any *demands*, tweak the global configuration or the repository configuration. This method is called immediately after the CLI/extension is finished configuring DNF.

**run**()
Run the command. This method is invoked by the CLI when this command is executed. Should raise

*dnf.exceptions.Error* with a proper message if the command fails. Otherwise should return `None`. Custom commands typically override this method and put their main work code here.

**class** `dnf.cli.`**Cli**

Manages the CLI, including reading configuration, parsing the command line and running commands.

**demands**

An instance of *DemandSheet*, exposed to allow custom commands and plugins influence how the CLI will operate.

**register_command(command_cls):**

Register new command. *command_cls* is a subclass of *Command*.

**redirect_logger(self, stdout=None, stderr=None):**

Change minimal logger level for terminal output to stdout and stderr according to specific command requirements. For stdout and stderr use logging.INFO, logging.WARNING, etc.

Indices:

- genindex

CHAPTER 6

# DNF User's FAQ

**Contents**

# General Questions

## What does DNF stand for?

Dandified Yum.

## Can I have DNF and Yum installed side by side?

Yes, you can. And this setup is tested by many.

There is one restriction: DNF and Yum keep additional data about each installed package and every performed transaction. This data is currently not shared between the two managers so if the admin installs half of the packages with DNF and the other half with Yum then each program can not benefit from the information held by the other one. The practical bottom line is that commands like `autoremove` can not take a completely informed decision and thus have to "play it safe" and remove only a subset of dependencies they would be able to otherwise. Similar situation exists with groups.

To transfer transaction additional data from yum to DNF, run:

```
dnf install python-dnf-plugins-extras-migrate && dnf-2 migrate
```

## Is there a compatibility layer for Yum?

For the CLI, yes. Just install `dnf-yum` which supplies our own `/usr/bin/yum`. Note two things: all the *differences* between the two package managers still apply and this does not provide "yum" in terms of package dependencies (it conflicts with the Yum package though).

## What to do with packages that DNF refuses to remove because their `%pre` or `%preun` scripts are failing?

If this happens, it is a packaging error and consider reporting the failure to the package's maintainer.

You can usually remove such package with `rpm`:

```
rpm -e <package-version> --noscripts
```

## Why are `dnf check-update` packages not marked for upgrade in the following `dnf upgrade`

Sometimes one can see that a newer version of a package is available in the repos:

```
$ dnf check-update
libocsync0.x86_64 0.91.4-2.1              devel_repo
owncloud-client.x86_64 1.5.0-18.1         devel_repo
```

Yet the immediately following `dnf upgrade` does not offer them for upgrade:

```
$ dnf upgrade
Resolving dependencies
--> Starting dependency resolution
--> Finished dependency resolution
```

```
Dependencies resolved.
Nothing to do.
```

It might seem odd but in fact this can happen quite easily: what the first command does is only check whether there are some available packages with the same name as an installed package but with a higher version. Those are considered upgrade candidates by `check-update`, but no actual dependency resolving takes place there. That only happens during `dnf upgrade` and if the resolving procedure then discovers that some of the packages do not have their dependencies ready yet, then they are not offered in the upgrade. To see the precise reason why it was not possible to do the upgrade in this case, use:

```
$ dnf upgrade --best
```

In DNF version 1.1 and above, you can see the skipped packages in the special transaction summary section. In order to pull these packages into transaction one has to remove conflicting packages, to do that execute:

```
$ dnf upgrade --best --allowerasing
```

## Why do I get different results with `dnf upgrade` vs `yum update`?

We get this reported as a bug quite often, but it usually is not. One reason to see this is that DNF does not list update candidates as it explores them. More frequently however the reporter means actual difference in the proposed transaction. This is most often because the metadata the two packagers are working with were taken at a different time (DNF has a notoriously looser schedule on metadata updates to save time and bandwidth), and sometimes also because the depsolvers inside are designed to take a different course of action when encountering some specific update scenario.

The bottom line is: unless a real update problem occurs (i.e. DNF refuses to update a package that Yum updates) with the same set of metadata, this is not an issue.

## Is it possible to force DNF to get the latest metadata on `dnf upgrade`?

Yes, clear the cache first:

```
$ dnf clean metadata
$ dnf upgrade
```

or by one command line simply put:

```
$ dnf upgrade --refresh
```

An alternative is to shorten the default expiry time of repos, for that edit `/etc/dnf/dnf.conf` and set:

```
metadata_expire=0
```

Of course, some repos might use a custom `metadata_expire` value, you'll currently have to change these manually too.

If you're the kind of the user who always wants the freshest metadata possible, you'll probably want to *disable the automatic MD updates*.

## How do I disable automatic metadata synchronization service?

Several ways to do that. The DNF way is to add the following to `/etc/dnf/dnf.conf`:

```
metadata_timer_sync=0
```

### Shouldn't DNF exit soon from certain commands if it is not run under root?

No, there can be systems and scenarios that allow other users than root to successfully perform `dnf install` and similar and it would be impractical to stop these from functioning by the UID check. Alternatively, the practice of checking filesystem permissions instead of the effective UID could lead to false positives since there is plenty of time between DNF startup and the possible transaction start when permissions can be changed by a different process.

If the time loss incurred by repeated runs of DNF is unacceptable for you, consider using the noroot plugin.

## Using DNF in Fedora

### For my stable Fedora release, can I install the rawhide packages for testing purposes?

Yes, in two steps: first install the necessary `.repo` files:

```
dnf install fedora-repos-rawhide
```

Then, when you want to include the packages from the rawhide repo, execute a DNF command with Rawhide enabled:

```
dnf --enablerepo=rawhide upgrade rpm
```

---

**Note:** Installing rawhide packages onto a stable Fedora release system is generally discouraged as it leads to less tested combinations of installed packages. Please consider this step carefully.

---

CHAPTER 7

# DNF Release Notes

## 2.7.3 Release Notes

Bugs fixed in 2.7.3:

- Bug 1472847 - dnf install dosn't handle 'Location:' http header
- Bug 1498426 - dnf download does not download RPMs in fedora rawhide container image
- Bug 1427144 - libselinux-python not installed when @ansible-node in kickstart %packages

## 2.7.2 Release Notes

API additions in 2.7.2:

- Added new option `--comment=<comment>` that adds a comment to transaction in history
- `dnf.Base.pre_configure_plugin()` configure plugins by running their pre_configure() method
- Added pre_configure() method for plugins and commands to configure dnf before repos are loaded

Bugs fixed in 2.7.2:

- Bug 1421478 - dnf repository-packages: error: unrecognized arguments: -x rust-rpm-macros
- Bug 1491560 - 'dnf check' reports spurious "has missing requires of" errors
- Bug 1465292 - DNF remove protected duplicate package
- Bug 1279001 - [RFE] Missing dnf –downloaddir option
- Bug 1212341 - [RFE] Allow plugins to override the core configuration
- Bug 1299482 - mock –init fails with message "Failed calculating RPMDB checksum"
- Bug 1192811 - dnf whatprovides should show which provides matched a pattern
- Bug 1288845 - "dnf provides" wildcard matching is unreliable (not all packages with matches listed)

- Bug 1237349 - dnf autoremove not removing what dnf list extras shows
- Bug 1470050 - the 'priority=' option in /etc/yum.repos.d/*.repo is not respected
- Bug 1347927 - dnf –cacheonly downloads packages
- Bug 1478115 - [abrt] dnf: _hcmd_undo(): __init__.py:888:_hcmd_undo:IndexError: list index out of range
- Bug 1461171 - RFE: support –advisory= with install
- Bug 1495116 - Dnf version fails with traceback in container
- Bug 1448874 - "dnf needs-restarting" vanished from bash completion

## 2.6.3 Release Notes

API additions in 2.6.3:

- Added auto substitution for all variables used for repo creation by *dnf.repodict.RepoDict.add_new_repo()*
- Added description of `--downloaddir=<path>` dnf option

Bugs fixed in 2.6.3:

- Bug 1476215 - repoquery –location sometimes prints invalid URL
- Bug 1473964 - dnf -C (or dnf –cacheonly) does not work as documented for a regular user
- Bug 1359482 - Trying to remove 'KDE Plasma Workspaces' group using dnf tries to remove systemd and dnf
- Bug 1476834 - [abrt] dnf: arch(): config.py:908:arch:TypeError: unhashable type: 'list'
- Bug 1244755 - [UX] additional dependencies to be installed or removed are not listed separately
- Bug 1476748 - Boltron: yum-dnf wrapper breaks installation of modules
- Bug 1476464 - dnf download broken if repository is file:/// (ie. local)
- Bug 1464192 - rpm.RPMTRANS_FLAG_TEST prevents to save newly imported gpg key
- Bug 1463107 - Undoing a remove transaction marks packages as user installed
- Bug 1426196 - RFE: Change "upgraded X packages" to "download X packages" when dnf download is used
- Bug 1457507 - [api] Explicitly set metalink is not substituted

## 2.6.2 Release Notes

API additions in 2.6.2:

- *dnf.conf.Conf.basearch*
- *dnf.conf.Conf.arch*
- *dnf.conf.Conf.ignorearch*
- Introduced new configuration option `autocheck_running_kernel`
- *dnf.subject.Subject.get_best_selector()* can use three additional key words: `obsoletes`, `reports`, and `reponame`.

From commandline it is possible to use new option `--noautoremove` to disable removal of dependencies that are no longer used.

Bugs fixed in 2.6.2:

- Bug 1279001 - [RFE] Missing dnf –downloaddir option
- Bug 1397848 - UnicodeDecodeError: 'utf-8' codec can't decode byte 0xf3 in position 1: invalid continuation byte
- Bug 1361424 - [RFE] add cli option for clean_requirements_on_remove=no
- Bug 1387925 - RFE: Display all unsigned package errors instead of only one
- Bug 1332099 - Installing packages produces a backtrace, installation goes ok
- Bug 1470116 - /etc/dnf/automatic.conf configuration for command_email is ignored
- Bug 1161950 - [visual] output and progress bar not visible during package downloads specified by remote url
- Bug 1320254 - [RFE] Support short abbrevations of commands like install in DNF
- Bug 1424723 - doc: Hard to find documentation for weak dependencies
- Bug 1462486 - RFE: Add support for groupmember on repoquery
- Bug 1314405 - No checksum check
- Bug 1457368 - dnf repoquery –obsoletes does not work
- Bug 1339280 - [rfe] Case insensitive match hint
- Bug 1138978 - [rfe] maybe you meant?
- Bug 1423472 - dnf list updates should only print newest version
- Bug 1427365 - DNF (and thus anaconda, livemedia-creator...) skips packages that are missing or cannot be installed due to conflicts or dependency issues with no warning, error or information
- Bug 1398871 - dnf group remove removes packages from other groups, and explicitly installed packages
- Bug 1432312 - removing MATE Desktop would remove dnf and systemd

## 2.5.1 Release Notes

API additions in 2.5.1:

- *`dnf.Plugin.pre_transaction()`* is a hook that is called just before transaction execution.
- *`dnf.subject.Subject.get_nevra_possibilities()`* returns generator for every possible nevra.

Bugs fixed in 2.5.1:

- Bug 1456419 - dnf update –refresh fails for repo_gpgcheck=1
- Bug 1445021 - Unlocalized date format
- Bug 1400714 - implement alwaysprompt=no option
- Bug 1250702 - dnf.Base.group_install does not honor multilib_policy=all
- Bug 1381988 - [RFE] output excluded packages during update
- Bug 1397848 - UnicodeDecodeError: 'utf-8' codec can't decode byte 0xf3 in position 1: invalid continuation byte

- Bug 1321407 - ValueError: (2, 'LRO_MAXSPEED (400) is lower than LRO_LOWSPEEDLIMIT (1000)', 'Bad argument of a handle option')
- Bug 1291867 - [conn] show progress when repo has metalink
- Bug 1372895 - don't access network/load latest repo metadata when doing "list installed"
- Bug 1444751 - dnf repoquery allow –qf use with –tree

## 2.5.0 Release Notes

API additions in 2.5.0:

*dnf.callback.DownloadProgress.start()* can use one additional key word `total_drpms`.

Bugs fixed in 2.5.0:

- Bug 1350546 - print more useful info when package can not be installed
- Bug 1449618 - dnf won't explain conflicts even with –best
- Bug 1270451 - Don't show packages from lower priority repos which are masked/blocked by higher prority repo packages
- Bug 1254966 - [UX] dnf install tries to update the installed dependencies to newer version (at least message says that)
- Bug 1426787 - "add '–best' to command line to see why package dependency cannot be satisfied" doesn't help
- Bug 1293983 - [UX] show skipped packages during upgrade with pkg param
- Bug 1370062 - history userinstalled should not show NEVRA
- Bug 1293067 - RFE: dnf suggests –allowerasing even when it cannot help
- Bug 1393814 - dnf reports a lot of conflicting packages where there are no conflicts
- Bug 1398040 - dnf calls weak dependencies "Skipping packages with broken dependencies"
- Bug 1342157 - Search that does not match anything has odd "error" message
- Bug 1379906 - Traceback when dependency of plugin is removed and plugin calls the dependency
- Bug 1198975 - [UX] show progress for DRPM

## 2.4.1 Release Notes

DNF command additions in 2.4.1:

- `dnf [options] repoquery --userinstalled` limit the resulting set only to packages installed by user.

Bugs fixed in 2.4.1:

- Bug 1446756 - Msgid bug, the flag "supplements" is misspelled in a help text
- Bug 1446432 - Minimal install option install full gnome desktop
- Bug 1446641 - dnf repoquery –duplicates –latest-limit -1 -q does not work
- Bug 1278124 - RFE: Add new option to repoquery - user-installed
- Bug 1301868 - dnf search with multiple matches shows repeated grouping

## 2.4.0 Release Notes

API additions in 2.4.0:

- *dnf.subject.Subject.get_best_query()* can use two additional key words: with_nevra, and with_filenames.
- Added description of *dnf.repo.Repo.cost*
- Added description of *dnf.repo.Repo.excludepkgs*
- Added description of *dnf.repo.Repo.includepkgs*

DNF command additions in 2.4.0:

- --enableplugin=<plugin names> *command line argument* enable the listed plugins specified by names or globs.
- --releasever=<release> *command line argument* now autodetect releasever in installroot from host if / value is used as <release>.

Bugs fixed in 2.4.0:

- Bug 1302935 - Untranslatable element in the header
- Bug 1248684 - RFE: add exclude and cost properties to dnf.repo.Repo API
- Bug 1441636 - –installroot against non-existent rpm database no longer works
- Bug 1438438 - AssertionError from DNF
- Bug 1256313 - DNF should download packages specified by URL into the cache directory
- Bug 1161950 - [visual] output and progress bar not visible during package downloads specified by remote url
- Bug 1421244 - DNF keeps packages with broken scriptlets installed, although it claims they were removed.

## 2.3.0 Release Notes

API additions in 2.3.0:

- *dnf.package.Package.remote_location()* returns location from where the package can be downloaded from.

DNF command additions in 2.3.0:

- dnf [options] repoquery --whatconflicts <capability> limit the resulting set only to packages that conflict <capability>.
- dnf [options] repoquery --whatobsoletes <capability> limit the resulting set only to packages that obsolete <capability>.
- dnf [options] repoquery --location show a location where the package could be downloaded from.
- dnf [options] repoquery --nvr show found packages in format name-version-release.
- dnf [options] repoquery --nevra show found packages in format name-epoch:version-release.architecture (default).
- dnf [options] repoquery --envra show found packages in format epoch:name-version-release.architecture.

- `dnf [options] repoquery --recursive` query packages recursively. Can be used with `--whatrequires <REQ>` (optionaly with –alldeps, but it has no effect with –exactdeps), or with `--requires <REQ> --resolve`.

Bugs fixed in 2.3.0:

- Bug 1290137 - RFE: dnf repoquery list not all available tags and there no full support of repoquery –location
- Bug 1349314 - Dnf autoremove removes kernel-modules-extra but dnf update reinstalls those kernel modules again.
- Bug 1247122 - implement –randomwait option
- Bug 1298717 - Translation missing, when package already downloaded.

## 2.2.0 Release Notes

API additions in 2.2.0:

- *`dnf.callback.TransactionProgress.progress()`* has new actions: TRANS_PREPARATION, TRANS_POST, and PKG_SCRIPTLET.

Bugs fixed in 2.2.0:

- Bug 1411432 - doc: Expected order of actions not documented for dnf.callback.TransactionProgress
- Bug 1406130 - Upgrade does not show progress between Cleanup and Verify
- Bug 1411423 - RFE: Add package scriptlet action to dnf.callback.TransactionProgress
- Bug 1369212 - dnf repolist with no repos

## 2.1.1 Release Notes

Bugs fixed in 2.1.1:

- Bug 1417542 - Dnf won't synchronize cache
- Bug 1401446 - dnf does not report cleaning packages
- Bug 1416699 - Error: Will not install a source rpm package
- Bug 1427132 - create an API for initializing repos
- Bug 1397047 - [abrt] dnf: _scriptError(): rpmtrans.py:550:_scriptError:AttributeError: 'NoneType' object has no attribute 'name'
- Bug 1379628 - [RFE] Add "–nodocs" option for install
- Bug 1424939 - L10N - explain "Login User"
- Bug 1396992 - dnf repoquery/whatrequires doesn't work correctly
- Bug 1412970 - [RFE] repoquery: Allow querying all packages

## 2.1.0 Release Notes

API additions in 2.1.0:

- *dnf.Base.update_cache()* downloads and caches in binary format metadata for all known repos.

Bugs fixed in 2.1.0:

- Bug 1421835 - dnf list command broken
- Bug 1415711 - DNF should not automatically download repodata over metered connections
- Bug 1417627 - RFE: show hint how to display why package was skipped

## 2.0.1 Release Notes

API changes in 2.0.1:

- *dnf.Base.package_downgrade()* now accept keyword strict to ignore problems with dep-solving

API additions in 2.0.1:

- *dnf.Base.autoremove()* removes all 'leaf' packages from the system that were originally installed as dependencies
- `dnf.cli.Cli.redirect_logger()` changes minimal logger level for terminal output to stdout and stderr

DNF command additions in 2.0.1:

- `dnf [options] shell [filename]` opens an interactive shell for conducting multiple commands during a single execution of DNF
- `dnf [options] swap <remove-spec> <install-spec>` removes spec and install spec in one transaction

Bugs fixed in 2.0.1:

- Bug 1409361 - [abrt] dnf: output.py:1508:historyInfoCmd:AttributeError: 'list' object has no attribute 'add'
- Bug 1414512 - [rfe] Add dnf api for autoremove
- Bug 1238808 - [UX] Negative percentage reported when delta rebuilds fail
- Bug 1386085 - L10N - Uncut sentence DepSolveProgressCallBack > pkg_added
- Bug 1286553 - [UX] Should say "Freed space" rather than "Installed size" when removing packages
- Bug 1337731 - Live image compose fails with dnf 1.1.9
- Bug 1336879 - dnf fails to respect conditional level in comps.xml
- Bug 1173349 - wildcard doesn't work properly when downgrade
- Bug 1329617 - Downgrade fails with broken dep with –setopt=strict=False
- Bug 1283255 - downgrade doesn't resolve dependencies automatically
- Bug 1369411 - information about metadata expiration should go into stderr
- Bug 1243393 - Query.filter(pkgs=XXX) is not documented
- Bug 1243393 - Query.filter(pkgs=XXX) is not documented
- Bug 1411349 - dnf makes wrong assumptions about metalinks
- Bug 1345976 - [UX] When no repo and –disablerepo=* - incorrect error message
- Bug 1369212 - dnf repolist with no repos
- Bug 1349247 - [doc] Document dnf grouplist hidden

- Bug 1403930 - dnf repoquery -f with multiple files doesn't seem to work
- Bug 1403465 - [RFE] Transaction modifiers for "dnf install" and "dnf remove"
- Bug 1110780 - dnf swap not supported
- Bug 1405333 - [RFE] add history to prompt line in dnf shell
- Bug 1254879 - dnf repoquery -l should display '(contains no files)' if there are no files

## 2.0.0 Release Notes

List of all incompatible changes can be found at: *dnf-1 vs dnf-2*

API changes in 2.0.0:

- *dnf.Base.add_remote_rpms()* now suppresses any error if `strict` equals to `False`.
- *dnf.Base.read_comps()* now limits results to system basearch if `arch_filter` equals to `True`.
- `dnf.cli.Cli.configure()` now doesn't take any additional arguments.
- `dnf.cli.Cli.run()` now doesn't take any additional arguments.
- *dnf.Plugin.read_config()* now doesn't take any name of config file.
- `dnf.Repo.__init__()` now takes *parent_conf* argument which is an instance of *dnf.conf.Conf* holding main dnf configuration instead of *cachedir* path.
- `exclude` and `include` configuration options change to `excludepkgs` and `includepkgs`.

API additions in 2.0.0:

- *dnf.Base.init_plugins()* initializes plugins. It is possible to disable some plugins by passing the list of their name patterns to `disabled_glob`.
- *dnf.Base.configure_plugins()* configures plugins by running their `configure()` method.
- `dnf.Base.urlopen()` opens the specified absolute `url` and returns a file object which respects proxy setting even for non-repo downloads
- Introduced new configuration options: `check_config_file_age`, `clean_requirements_on_remove`, `deltarpm_percentage`, `exit_on_lock`, `get_reposdir`, `group_package_types`, `installonlypkgs`, `keepcache`, `protected_packages`, `retries`, `type`, and `upgrade_group_objects_upgrade`. For detailed description see: *DNF API*.
- Introduced new configuration methods: `dump()` and `write_raw_configfile()`. For detailed description see: *DNF API*.
- Introduced *dnf.package.Package* attributes `debug_name`, `downloadsize`, `source_debug_name` and `source_name`. For detailed description see: *DNF Package API*.
- *dnf.query.Query.extras()* returns a new query that limits the result to installed packages that are not present in any repo.
- `dnf.repo.Repo.enable_debug_repos()` enables debug repos corresponding to already enabled binary repos.
- `dnf.repo.Repo.enable_source_repos()` enables source repos corresponding to already enabled binary repos.
- *dnf.repo.Repo.dump()* prints repository configuration, including inherited values.
- *dnf.query.Query.filter()* now accepts optional argument *pkg*.

DNF command changes in 2.0.0:

- `dnf [options] group install [with-optional] <group-spec>...` changes to `dnf [options] group install [--with-optional] <group-spec>...`.
- `dnf [options] list command [<package-name-specs>...]` changes to *dnf [options] list –command [<package-name-specs>...]*'.
- `dnf [options] makecache timer` changes to `dnf [options] makecache --timer`.
- `dnf [options] repolist [enabled|disabled|all]` changes to `dnf [options] repolist [--enabled|--disabled|--all]`.
- `dnf [options] repository-packages <repoid> info command [<package-name-spec>...]` changes to `dnf [options] repository-packages <repoid> info --command [<package-name-spec>...]`.
- `dnf repoquery --duplicated` changes to `dnf repoquery --duplicates`.
- `dnf [options] search [all] <keywords>...` changes to `dnf [options] search [--all] <keywords>...`.
- `dnf [options] updateinfo [<availability>] [<spec>...]` changes to `dnf [options] updateinfo [--summary|--list|--info] [<availability>] [<spec>...]`.
- `--disablerepo` *command line argument* is mutually exclusive with `--repo`.
- `--enablerepo` *command line argument* now appends repositories.
- `--installroot` *command line argument*. For detailed description see: *DNF command API*.
- `--releasever` *command line argument* now doesn't detect release number from running system.
- `--repofrompath` *command line argument* can now be combined with `--repo` instead of `--enablerepo`.
- Alternative of yum's `deplist` changes from `dnf repoquery --requires` to `dnf repoquery --deplist`.
- New systemd units *dnf-automatic-notifyonly*, *dnf-automatic-download*, *dnf-automatic-download*

were added for a better customizability of *dnf-automatic*.

DNF command additions in 2.0.0:

- `dnf [options] remove --duplicates` removes older version of duplicated packages.
- `dnf [options] remove --oldinstallonly``removes old installonly packages keeping only ``installonly_limit` latest versions.
- `dnf [options] repoquery [<select-options>] [<query-options>] [<pkg-spec>]` searches the available DNF repositories for selected packages and displays the requested information about them. It is an equivalent of `rpm -q` for remote repositories.
- `dnf [options] repoquery --querytags` provides list of recognized tags by repoquery option *--queryformat*.
- `--repo` *command line argument* enables just specific repositories by an id or a glob. Can be used multiple times with accumulative effect. It is basically shortcut for `--disablerepo="*" --enablerepo=<repoid>` and is mutually exclusive with `--disablerepo` option.
- New commands have been introduced: `check` and `upgrade-minimal`.
- New security options introduced: `bugfix`, `enhancement`, `newpackage`, `security`, `advisory`, `bzs`, `cves`, `sec-severity` and `secseverity`.

Bugs fixed in 2.0.0:

- Bug 1229730 - dnf.subject.Subject() does not honour epoch
- Bug 1375277 - releasever in baseurl translated to major.minor, not just major (7.3 instead of 7, yum incompatibility)
- Bug 1384289 - dnf repoquery –quiet outputs “Last metadata expiration check:”
- Bug 1398272 - Dnf import of public keys causes stale rpmdb locks left behind
- Bug 1382224 - RFE: dnf transactions should run in a transient systemd service
- Bug 1177785 - add hawkey.Selector epoch support
- Bug 1272109 - Dnf very slow with wildcards.
- Bug 1234930 - [RFE] Add yum-security functionality to DNF
- Bug 1341086 - dnf install exists with 0 even if it did not install all packages
- Bug 1382247 - installonlypkgs doesn’t work by provides
- Bug 1381216 - –nogpgcheck doesn’t work properly
- Bug 1381432 - L10N - update DNF 2.0 strings in Zanata so we can translate and fix tiny localization issues
- Bug 1096506 - promoting ‘install a’ to ‘install b’ when b obsoletes a
- Bug 1332830 - DNF builddep is not consistently installing packages providing capabilities properly
- Bug 1348766 - “The operation would result in removing the booted kernel” when installonlypkgs is set in dnf.conf
- Bug 1337731 - Live image compose fails with dnf 1.1.9
- Bug 1333591 - F24 PAE kernel conflicts with F23 PAE kernel
- Bug 1314961 - dnf-automatic crashes when ‘None’ emitter is used
- Bug 1372307 - dnf-automatic does not run etckeeper’s autocommit
- Bug 1373108 - Whitelist ‘type=*’ option in config
- Bug 1148627 - dnf doesn’t provide enough info to understand the package dependencies
- Bug 1267298 - dnf doesn’t handle uncompressed comps.xml
- Bug 1373591 - TypeError: decode() argument 1 must be string, not MagicMock
- Bug 1230355 - Running “dnf history” as non-root user produces unhelpful error message
- Bug 1366793 - Nothing obsoletes retired dnf-langpacks packages, breaks upgrade from Fedora 23 to 25+
- Bug 1369411 - information about metadata expiration should go into stderr
- Bug 1366793 - Nothing obsoletes retired dnf-langpacks packages, breaks upgrade from Fedora 23 to 25+
- Bug 1369459 - dnf repolist disabled doesn’t show repository name
- Bug 1306096 - time zone not displayed
- Bug 1368832 - AttributeError: ‘Repo’ object has no attribute ‘metalink_path’
- Bug 1366793 - Nothing obsoletes retired dnf-langpacks packages, breaks upgrade from Fedora 23 to 25+
- Bug 1359016 - Unknown configuration option: enabled_metadata = 1
- Bug 1365593 - dnf: misleading error message when trying to “upgrade” src.rpm
- Bug 1297087 - [RFE] Implementation of localpkg_gpgcheck option for checking gpg signature for local packages or packages specified by URL

- Bug 1227053 - '–exclude' option empties whole transaction
- Bug 1356926 - Results of 'dnf repoquery –whatprovides' and 'dnf provides' don't match
- Bug 1055910 - [rfe] includepkgs directive support
- Bug 1219867 - include option in repository should not effect the rest repos
- Bug 1226677 - dnf repoquery -f does not support wildcards
- Bug 1350604 - RFE: Add 'build-dep' and 'distrosync' aliases to 'builddep' and 'distro-sync'
- Bug 1253120 - dnf remove: when remove multiple versions of kernel packages, the "Erasing" result of "Running transaction" show duplicated lowest version number
- Bug 1158548 - [rfe] "dnf <plugin> –help" does not print plugin help
- Bug 1262878 - dnf –exclude doesn't work for local installed package
- Bug 1318852 - DNF doesn't install groups, reports the group is already installed.
- Bug 1327438 - make –available default in dnf repoquery
- Bug 1343880 - dnf.plugin.repoquery documentation should use/explain –alldeps
- Bug 1338921 - dnf removes sqlite-libs because plexmediaserver pretends it provides them
- Bug 1284349 - installed hipchat confuses dnf to remove sqlite (which destroys the system)
- Bug 1338921 - dnf removes sqlite-libs because plexmediaserver pretends it provides them
- Bug 1284349 - installed hipchat confuses dnf to remove sqlite (which destroys the system)
- Bug 1306096 - time zone not displayed
- Bug 1218071 - No API to enable/run plugins
- Bug 1193823 - [RFE] Plugins can't be disabled in config files
- Bug 1246211 - DNF Repoquery: Automatically enable repo selected by –repo
- Bug 1193851 - [RFE] Add support for –repoid parameter
- Bug 1158548 - [rfe] "dnf <plugin> –help" does not print plugin help
- Bug 1215208 - Double-dash ("–") argument separator is not supported
- Bug 1212693 - dnf does not consistently exit immediately when an invalid option is given
- Bug 1212341 - [RFE] Allow plugins to override the core configuration
- Bug 1306591 - when weak_install_weak_deps==False don't show weak dependencies in "Skipping packages with conflicts" section
- Bug 1227001 - Using DNF to with –installroot fails where yum works
- Bug 1163028 - [doc] Document behaviour of –installroot
- Bug 1279185 - dnf searches in /etc/yum.repos.d/*.repo when using –installroot
- Bug 1289067 - [RFE][UX] visual separation of autoremoved packages
- Bug 1328674 - diskspacecheck=0 in dnf.conf has no effect
- Bug 1380580 - dnf 2.0 doesn't accept type=rpm in repo files
- Bug 1327999 - "dnf upgrade-to" requiring full NEVRA
- Bug 1400081 - Crash with empty includepkgs list in repository configuration file
- Bug 1293782 - dnf clean all does not wait for current running upgrade

- Bug 1386078 - L10N - Explain repoquery strings
- Bug 1358245 - repoquery: valid use of stderr and stdout, use stdout only for requested data
- Bug 1243393 - Query.filter(pkgs=XXX) is not documented
- Bug 1339739 - Option –skip-broken is missing

## 1.1.10 Release Notes

Fixed unicode handling and fixing other bugs.

Bugs fixed in 1.1.10:

- Bug 1257965 - DNF doesn't treat the "xml:base" attribute in the primary.xml location tag correctly
- Bug 1352130 - dnf should not report errors to stdout, it's mildly dangerous
- Bug 1343764 - dnf reinstall deletes yumdb entry
- Bug 1308994 - After installing a package from local unicode directory a traceback is shown (UnicodeEncodeError)
- Bug 1230183 - Package update does not work when package changes from noarch <-> arch
- Bug 1295090 - [abrt] dnf: i18n.py:44:write:UnicodeEncodeError: 'ascii' codec can't encode character '\xe6' in position 54: ordinal not in range(128)
- Bug 1325869 - [abrt] dnf: config.py:91:__set__:ValueError: Error parsing "baseurl = repodata": URL must be http, ftp, file or https not ""
- Bug 1338046 - dnf will consume 100% CPU when there is no space left on device
- Bug 1214768 - history userinstalled lists should print NEVRAs instead of names
- Bug 1338504 - dnf upgrade (to 1.1.9) broke bash completion
- Bug 1338564 - Update fails with "AttributeError: 'Base' object has no attribute 'sigCheckPkg' (36)"

## 1.1.9 Release Notes

From this release if you use any non-API methods warning will be printed and bugfixes.

Bugs fixed in 1.1.9:

- Bug 1324086 - dnf "autoremove" command has no description in dnf –help
- Bug 1332012 - UnicodeEncodeError: 'ascii' codec can't encode characters in position 342-343: ordinal not in range(128)
- Bug 1292892 - dnf groupinstall ignores errors
- Bug 1328674 - diskspacecheck=0 in dnf.conf has no effect
- Bug 1286556 - /usr/lib64/python3.5/re.py:203: FutureWarning: split() requires a non-empty pattern match.
- Bug 1245121 - TypeError: unhashable type: 'YumHistoryPackageState'

## 1.1.8 Release Notes

Improvements in documentation, bugfixes, translation updates.

Bugs fixed in 1.1.8:

- Bug 1309408 - [RFE] Add docs that –best is showing more details on broken package dependencies
- Bug 1209649 - [UX] dnf group info (and other two word commands) give wrong mini_usage info
- Bug 1272977 - dnf waits 10 minutes for a single repo (for unresponsive ftp server?)
- Bug 1322226 - [pt-BR] some strings translated in zanata are missing in the terminal
- Bug 1315349 - dnf-makecache.service fills journal with useless lines because of verbose option
- Bug 1214562 - Config error: Error parsing "installonly_limit = u'1'": Out of range integer value.
- Bug 1313215 - got "IndexError: list index out of range " after ran dnf history command
- Bug 1306057 - [UX] different behaviour of installonly in yum/dnf
- Bug 1289164 - [RFE] positive prompt answer by default

## 1.1.7 Release Notes

Added *dnf.rpm.basearch()* method, intended for the detection of CPU base architecture.

The *group list* command was enriched with `installed` and `available` switches.

Documented a standard way of overriding autodetected arhitectures in *DNF API*.

Bugs fixed in 1.1.7:

- Bug 1286477 - DNF creates /etc/yum/repos.d instead of /etc/dnf/repos.d when no repodir is present/configured
- Bug 1305356 - dnf groupinstall does not install packages, only marks them
- Bug 1258503 - regression in dnf, requires network access for history
- Bug 1283432 - dnf group list –installed / –available
- Bug 1268818 - dnf -v group list not mentioned in man page
- Bug 1306304 - [perf] cache installed set of packages in query (for updates)
- Bug 1302934 - Malformed translations in 1.1.6
- Bug 1303149 - dnf history info failing
- Bug 1302217 - dnf metadata expiration message does not fit on screen

## 1.1.6 Release Notes

Added support of socks5 proxy.

Bugs fixed in 1.1.6:

- Bug 1291895 - DNF history failure if no history
- Bug 1256587 - The proxy option in dnf.conf doesn't support socks5
- Bug 1287221 - dnf stutters when writing in /var/log/dnf.rpm.log

- Bug 1277360 - doc: update specifying the packages in commands
- Bug 1294241 - file /var/lib/dnf/* is not owned by any package
- Bug 1289166 - dnf makecache triggers a python exception
- Bug 1294355 - German translation incomplete on delta RPMS
- Bug 1226322 - dnf clean all does not clean package metadata for local repos
- Bug 1275878 - [abrt] dnf: rpmsack.py:288:_delete:AttributeError: Cannot delete attribute installed_by on <dnf.yum.rpmsack.RPMDBAdditionalDataPackage object at 0x7f10c32f15d0>
- Bug 1239274 - dnf history does not show command line

## 1.1.5 Release Notes

Improved the start-up time of bash completion.

Reviewed documentation.

Bugs fixed in 1.1.5:

- Bug 1286619 - [abrt] dnf: group.py:253:_mark_install:AttributeError: 'GroupPersistor' object has no attribute 'environement'
- Bug 1229046 - mark group as installed only after successful transaction
- Bug 1282250 - RFE: don't consider remove operations as successful transactions
- Bug 1265391 - 'dnf update @group' doesn't work
- Bug 1283017 - [rfe] support braces in variable substitution
- Bug 1278592 - Bad default for email_from in automatic.conf
- Bug 1260421 - [rfe] dnf should download drpms first
- Bug 1278382 - DNF python programming error when run from virt-builder
- Bug 1230820 - /usr/bin/dnf using -OO
- Bug 1280240 - dnf should warn if plugins are disabled and an unknown command is requested

## 1.1.4 Release Notes

API additions in 1.1.4:

- newly added `dnf.Query.duplicated()`
- extended `dnf.Query.latest()`

Bugs fixed in 1.1.4:

- Bug 1278031 - dnf fails to find system-upgrade plugin with certain locales (Chinese, Japanese and maybe more)
- Bug 1264032 - RFE: add "Skipped" into Transaction Summary
- Bug 1209056 - [UX] [TRANS] Plural strings are not extracted properly
- Bug 1274946 - [abrt] dnf: main.py:99:main:IOError: [Errno 5] Input/output error

## 1.1.3 Release Notes

Now *dnf.Base.group_install()* is able to exclude mandatory packages of the group from transaction.

## 1.1.2 Release Notes

Implemented *–downloadonly* command line option.

Bugs fixed in 1.1.2:

- Bug 1262082 - 'ascii' codec can't decode byte 0xc3 in position 40: ordinal not in range(128)
- Bug 1250038 - [abrt] python-dnf: completion_helper.py:84:run:IndexError: list index out of range
- Bug 1048433 - RFE –downloadonly
- Bug 1259650 - "dnf provides" or "dnf repoquery –whatprovides" does not support globs
- Bug 1260198 - python-dnf should provide python2-dnf
- Bug 1259657 - No dnf-command() provides for dnf commands
- Bug 1254982 - dnf-automatic generates emails with broken charset
- Bug 1261766 - RFE: way to force no metadata expiration system-wide
- Bug 1234491 - [UX] dnf list installed: show repo id the package was installed from
- Bug 1256531 - dnf install crashes if terminal window is too small
- Bug 1254687 - DNF does not display any translations
- Bug 1261656 - DNF group update doesn't work
- Bug 1258364 - Some important dnf output goes to STDERR and not to STDOUT

## 1.1.1 Release Notes

Implemented `dnf mark` *command*.

Bugs fixed in 1.1.1:

- Bug 1249319 - dnf says "downloaded packages were saved in cache" before actually downloading any packages
- Bug 1234763 - wildcard in dnf update/downgrade should match installed packages only
- Bug 1242946 - [depsolv] dnf prefers package names over provide names
- Bug 1225225 - [abrt] python-dnf: repo.py:774:load:RepoError: Cache-only enabled but no cache for 'fedora'
- Bug 1254687 - DNF does not display any translations
- Bug 1247766 - dnf logrotate.d rotates logs too fast in many cases
- Bug 1125925 - [RFE] provide a 'add to user-installed-list' command
- Bug 1210289 - RFE: Have a way to wait on dnf lock rather than terminate

## 1.1.0 Release Notes

API additions in 1.1.0:

*dnf.Base.do_transaction()* now accepts multiple displays.

Introduced `install_weak_deps` *configuration* option.

Implemented `strict` *configuration* option.

API deprecations in 1.1.0:

- `dnf.callback.LoggingTransactionDisplay` is deprecated now. It was considered part of API despite the fact that it has never been documented. Use *dnf.callback.TransactionProgress* instead.

Bugs fixed in 1.1.0

- Bug 1210445 - [rfe] report when skipping available updates due to dependency reasons
- Bug 1218401 - [doc] document history_list_view and set it to cmd by default
- Bug 1227952 - dnf update on a local directory does not handle mutliple versions of the same package
- Bug 1197456 - [RFE] implement 'strict' config option (defaults=True)
- Bug 1236310 - dnf –quiet repolist doesn't output anything
- Bug 1219638 - DNF doesn't use xml:base in repodata
- Bug 1207981 - [UX] error during unpacking archive does not fail the installation
- Bug 1208918 - RFE: dnf shouldn't stop processing the command line at the first unknown package
- Bug 1221635 - add dnf config option to (not)obey weak deps
- Bug 1236306 - Some of dnf's subcommands can't handle pipe
- Bug 1234639 - dnf package actions triggered by dnf-automatic aren't logged to dnf.rpm.log
- Bug 1244486 - [doc] Add yum vs dnf hooks information in dnf documentation
- Bug 1224248 - [UX] Error message doesn't show names of packages well
- Bug 1243501 - 'dnf downgrade' treats installed packages as not installed
- Bug 1225237 - [UX] [doc] FILES section missing from manual pages

## 1.0.2 Release Notes

When a transaction is not successfully finished, DNF preserves downloaded packages until the next successful transaction even if `keepcache` option is set to `False`.

Maximum number of simultaneous package downloads can be adjusted by newly added `max_parallel_downloads` *configuration* option.

`--repofrompath` *command line argument* was introduced for temporary configuration of repositories.

API additions in 1.0.2:

Newly added package attributes: *dnf.package.Package.obsoletes*, *dnf.package.Package.provides* and *dnf.package.Package.requires*.

`dnf.package.Query.filter`'s keys `requires` and `provides` now accepts list of `Hawkey.Reldep` type.

Bugs fixed in 1.0.2:

- Bug 1148630 - add –allowerasing hint for conflict problems
- Bug 1176351 - [RFC] print warning message when we're going to install not latest version
- Bug 1210445 - [rfe] report when skipping available updates due to dependency reasons
- Bug 1173107 - mock: use of "dnf –installroot ..." yields "releasever not given" error
- Bug 1219199 - Problem when using proxy authentication in dnf.conf
- Bug 1220040 - [abrt] python-dnf: completion_helper.py:92:run:UnboundLocalError: local variable 'pkgs' referenced before assignment
- Bug 1230975 - [rfe] configurable parallel downloads (like yum's max_connections)
- Bug 1232815 - 'dnf install @group' returns error code 1 if all packages are already installed
- Bug 1113384 - [rfe] repofrompath option to specify ad-hoc repo
- Bug 1133979 - yum/dnf –quiet option not working for check-update command
- Bug 1238958 - dnf-automatic fails to send mail
- Bug 1238252 - Typo in dnf man page
- Bug 1212320 - [metadata] dnf makecache never timeouts

## 1.0.1 Release Notes

DNF follows the Semantic Versioning as defined at http://semver.org/.

Documented SSL *configuration* and *repository* options.

Added virtual provides allowing installation of DNF commands by their name in the form of `dnf install dnf-command(name)`.

*dnf-automatic* now by default waits random interval between 0 and 300 seconds before any network communication is performed.

Bugs fixed in 1.0.1:

- Bug 1214968 - dnf reports installed group does not exist
- Bug 1222694 - [abrt] python-dnf: comps.py:86:install_or_skip:TypeError: unsupported operand type(s) for +=: 'int' and 'TransactionBunch'
- Bug 1225246 - yum2dnf manpage: too much underlining
- Bug 1213985 - [rfe] add equivalent of "random_sleep" to dnf-automatic
- Bug 1225277 - python(abi) = 3.3 is needed by (installed)
- Bug 1223932 - UnicodeEncodeError: 'ascii' codec can't encode character u'\xe1' in position 19: ordinal not in range(128)
- Bug 1223614 - dnf groupinstall fails and returns success
- Bug 1203661 - [CRY] sslclientcert and sslclientkey not supported
- Bug 1187741 - dnf should state if packages explicitly requested for update aren't installed

## 1.0.0 Release Notes

Improved documentation of YUM to DNF transition in *Changes in DNF CLI compared to Yum*.

*Auto remove command* does not remove *installonly* packages.

*Downgrade command* downgrades to specified package version if that is lower than currently installed one.

DNF now uses `dnf.repo.Repo.id` as a default value for `dnf.repo.Repo.name`.

Added support of repositories which use basic HTTP authentication.

API additions in 1.0.0:

*configuration* options *username* and *password* (HTTP authentication)

`dnf.repo.Repo.username` and `dnf.repo.Repo.password` (HTTP authentication)

Bugs fixed in 1.0.0:

- Bug 1215560 - [abrt] dnf: i18n.py:165:exact_width:TypeError: 'NoneType' object is not iterable
- Bug 1199648 - [UX] dnf groupinstall claims that installed group does not exist
- Bug 1208773 - [RFE] Define virtual provides for DNF commands and suggest installing the appropriate one if a command was not found
- Bug 1208018 - config-manager: enable won't work. TypeError: unorderable types: Repo() < Repo()
- Bug 1207861 - Python 3: historical transactions that had multiple problems causes tracebacks
- Bug 1201445 - "dnf autoremove" removes "user-installed" installonly packages installed using "dnf upgrade"
- Bug 1210275 - DNF doesn't support http authentication
- Bug 1191275 - dnf downgrade <target-nevra> fails
- Bug 1207965 - Improve yum deprecated message
- Bug 1215289 - bash completion file adds a shell alias for python

## 0.6.5 Release Notes

Python 3 version of DNF is now default in Fedora 23 and later.

yum-dnf package does not conflict with yum package.

*dnf erase* was deprecated in favor of *dnf remove*.

Extended documentation of handling non-existent packages and YUM to DNF transition in *Changes in DNF CLI compared to Yum*.

API additions in 0.6.5:

Newly added *pluginconfpath* option in *configuration*.

Exposed *skip_if_unavailable* attribute from *Repository Configuration*.

Documented *IOError* exception of method *fill_sack* from `dnf.Base`.

Bugs fixed in 0.6.5:

- Bug 1203151 - Duplicated output for already installed packages
- Bug 1187579 - dnf upgrade ./<TAB> doesn't autocomplete

- Bug 1185977 - [api] remove cached package file only after transaction()
- Bug 1195240 - dnf-automatic “how to use”
- Bug 1193914 - ‘help history’ does not correspond with supported commands
- Bug 1195385 - dnf regression: cannot resolve provides with slashes
- Bug 1160806 - DNF shoud not deprecate “remove” command
- Bug 1186710 - dnf –setopt exclude=XXX adds garbage to excludes
- Bug 1207726 - Update dnf and yum packages for proposed dnf default change
- Bug 1157233 - ‘dnf download’ cannot be run in parallel as non-root user
- Bug 1190671 - DNF ships bash completition file but lacks requirement on bash completion
- Bug 1191579 - python3-dnf cannot be used without python2
- Bug 1195325 - RFE: provide a way to configure the config directory used by plugins
- Bug 1154202 - [depsolv] removing a package results in downloading more packages
- Bug 1189083 - [doc] please document ‘skip_if_unavailable’ Repo flag
- Bug 1193915 - [abrt] dnf: output.py:1730:historySummaryCmd:ValueError: need more than 0 values to unpack
- Bug 1195661 - lock files cannot be created for non logged users
- Bug 1190458 - [abrt] dnf: cli.py:210:do_transaction:UnicodeEncodeError: ‘ascii’ codec can’t encode character u’\xf3’ in position 196: ordinal not in range(128)
- Bug 1194685 - CACHEDIR_SUFFIX appended twice to cachedir when using dnf CLI.
- Bug 1160950 - “dnf install foo.rpm” doesn’t upgrade “foo”

## 0.6.4 Release Notes

Added example code snippets into *DNF Use Cases*.

Shows ordered groups/environments by *display_order* tag from *cli* and *Comps, or the Distribution Compose Metadata* DNF API.

In commands the environment group is specified the same as *group*.

*skip_if_unavailable* configuration option affects the metadata only.

added *enablegroups*, *minrate* and *timeout* *configuration options*

API additions in 0.6.4:

Documented *install_set* and *remove_set attributes* from *Transaction*.

Exposed *downloadsize*, *files*, *installsize* attributes from *Package*.

Bugs fixed in 0.6.4:

- Bug 1155877 - [abrt] dnf: i18n.py:130:ucd:UnicodeDecodeError: ‘ascii’ codec can’t decode byte 0xc3 in position 1: ordinal not in range(128)
- Bug 1175466 - add timeout option to repo conf
- Bug 1175466 - add timeout option to repo conf
- Bug 1186461 - “file” filter in DNF

- Bug 1170156 - [RFE][UX] Display metadata timestamp
- Bug 1184943 - please enrich DNF API with a method for setting up base.conf.cachedir for non privileged users
- Bug 1177002 - comps display_order for environment groups not respected by installer with dnf payload (or dnf itself)
- Bug 1169165 - let glob pattern describing package to match multiple packages
- Bug 1167982 - dnf.conf man page is in the wrong section
- Bug 1157233 - 'dnf download' cannot be run in parallel as non-root user
- Bug 1138096 - [doc] add examples on install, resolve, do_transaction api usage
- Bug 1181189 - dnf downgrade ./<TAB> doesn't autocomplete
- Bug 1181397 - [abrt] dnf: output.py:683:_display_packages_verbose:KeyError: u'ppc64-utils'
- Bug 1175434 - Odd placement and no rotation of hawkey.log
- Bug 1162887 - [api] introspecting transaction package sets
- Bug 1156084 - dnf install @some-environment-id does not work
- Bug 1175098 - dnf: unowned dir /etc/dnf/protected.d
- Bug 1174136 - wrongly computed text width with nonascii characters
- Bug 1055910 - [rfe] includepkgs directive support
- Bug 1155918 - let "history list" show all records
- Bug 1119030 - skip_if_unavailable to correctly treat repos that are down during package download
- Bug 1177394 - Dnf specifies both "size" and an age in the logrotate configuration
- Bug 1154476 - [abrt] dnf: lock.py:64:_read_lock:ValueError: invalid literal for int() with base 10: ''

## 0.6.3 Release Notes

*Deltarpm* configuration option is set on by default.

API additions in 0.6.3:

- dnf-automatic adds *motd emitter* as an alternative output

Bugs fixed in 0.6.3:

- Bug 1153543 - dnf bash completion hangs when trying to complete local path
- Bug 1151231 - Broken bash completion (does not occur when dnf-plugins-core installed)
- Bug 1163063 - dnf install with glob pattern output should be similar to yum install with glob
- Bug 1151854 - –cacheonly is not being respected – will update the cache if expired
- Bug 1151740 - dnf base api should not use errorSummary from dnf.cli.BaseCli
- Bug 1110780 - dnf swap not supported
- Bug 1149972 - update command implicitly working as reinstall
- Bug 1150474 - dnf group install broken for locale other than en_US
- Bug 995537 - RFE: /etc/motd emitter for yum-cron
- Bug 1149952 - Python exception thrown when searching dnf history

- Bug 1149350 - Odd placement and no rotation of hawkey.log
- Bug 1170232 - dnf eats newlines from error message
- Bug 1147523 - missing group info environment group support
- Bug 1148208 - [rfe] enable deltarpm support by default
- Bug 1109927 - RFE: take systemd inhibitor lock while doing operations that shouldn't be interrupted by shutdown

## 0.6.2 Release Notes

API additions in 0.6.2:

- Now `dnf.Base.package_install()` method ignores already installed packages
- *CliError* exception from `dnf.cli` documented
- *Autoerase*, *History*, *Info*, *List*, *Provides*, *Repolist* commands do not force a sync of expired *metadata*
- *Install* command does installation only

Bugs fixed in 0.6.2:

- Bug 909856 - [rfe] lazy mode for metadata syncing
- Bug 1134893 - "dnf update" works as "dnf install" on local package files
- Bug 1138700 - dnf doesn't install all packages on command line
- Bug 1070902 - dnf autocomplete for install ./* + <TAB> takes ages
- Bug 1124316 - [locking] per-installroot metadata lock
- Bug 1136584 - [abrt] dnf: rpmsack.py:255:_read:AttributeError: <dnf.yum.rpmsack.RPMDBAdditionalDataPackage object at 0x7f10fed01b50> has no attribute reason
- Bug 1135861 - group erase libreoffice fails, package requires but none of the providers can be installed
- Bug 1136223 - [abrt] dnf: packages.py:76:parsePackages:TypeError: unsupported operand type(s) for |=: 'set' and 'list'
- Bug 1122617 - reinstall of local rpms gives traceback
- Bug 1133830 - Does not import gpg keys
- Bug 1121184 - dnf: shows fake download speed when not downloading anything

## 0.6.1 Release Notes

New release adds *upgrade-type command* to *dnf-automatic* for choosing specific advisory type updates.

Implemented missing *history redo command* for repeating transactions.

Supports *gpgkey* repo config, *repo_gpgcheck* and *gpgcheck* [main] and Repo configs.

Distributing new package *dnf-yum* that provides */usr/bin/yum* as a symlink to */usr/bin/dnf*.

API additions in 0.6.1:

- *exclude*, the third parameter of `dnf.Base.group_install()` now also accepts glob patterns of package names.

Bugs fixed in 0.6.1:

- Bug 1132335 - RFE [UX] - Give users hint on uknown command
- Bug 1071854 - dnf reinstall does not work as expected
- Bug 1131969 - [rfe] group_install: accepts package specs as excludes
- Bug 908764 - RFE: Support repo-pkgs command set
- Bug 1130878 - [abrt] dnf: persistor.py:60:_diff_dcts:AttributeError: 'NoneType' object has no attribute 'keys'
- Bug 1130432 - [abrt] dnf: locale.py:166:_group:UnicodeDecodeError: 'ascii' codec can't decode byte 0xc2 in position 0: ordinal not in range(128)
- Bug 1118236 - dnf doesn't seem to support repo_gpgcheck config
- Bug 1109915 - [rfe] yum-cron replacement

## 0.6.0 Release Notes

0.6.0 marks a new minor version of DNF and the first release to support advisories listing with the *udpateinfo command*.

Support for the *include configuration directive* has been added. Its functionality reflects Yum's `includepkgs` but it has been renamed to make it consistent with the `exclude` setting.

Group operations now produce a list of proposed marking changes to group objects and the user is given a chance to accept or reject them just like with an ordinary package transaction.

Bugs fixed in 0.6.0:

- Bug 850912 - RFE: support updateinfo
- Bug 1055910 - [rfe] includepkgs directive support
- Bug 1116666 - handling groups installed by Yum
- Bug 1118272 - search command show traceback when in a debug mode
- Bug 1127206 - [abrt] dnf: __init__.py:186:install:Exception: Goal operation failed.

## 0.5.5 Release Notes

The full proxy configuration, API extensions and several bugfixes are provided in this release.

API changes in 0.5.5:

- *cachedir*, the second parameter of *dnf.repo.Repo.__init__()* is not optional (the method has always been this way but the documentation was not matching)

API additions in 0.5.5:

- extended description and an example provided for *dnf.Base.fill_sack()*
- *dnf.conf.Conf.proxy*
- *dnf.conf.Conf.proxy_username*
- *dnf.conf.Conf.proxy_password*
- *dnf.repo.Repo.proxy*

- *dnf.repo.Repo.proxy_username*
- *dnf.repo.Repo.proxy_password*

Bugs fixed in 0.5.5:

- Bug 1100946 - dnf install <bin filename> installs both i686 and x86_64 packages for no reason
- Bug 1117789 - [doc] [api] custom repos
- Bug 1120583 - [api] full-featured proxy configuration
- Bug 1121280 - History list with login names fails with non-ascii names
- Bug 1122900 - [abrt] dnf: package.py:73:header:RuntimeError: (Rpm file does not exist)
- Bug 1123688 - dnf missing Requires on pyliblzma

## 0.5.4 Release Notes

Several encodings bugs were fixed in this release, along with some packaging issues and updates to *DNF Configuration Reference*.

Repository *priority* configuration setting has been added, providing similar functionality to Yum Utils' Priorities plugin.

Bugs fixed in 0.5.4:

- Bug 1048973 - [rfe] repo priorities
- Bug 1108908 - dnf dies immediately complaining that ascii codec can't decode 0xc3
- Bug 1116544 - [i18n] double width Unicode characters
- Bug 1116839 - CompsError: Group 'Core' is already installed.
- Bug 1116845 - dnf crashed when executing 'dnf -v repolist all'
- Bug 1117102 - [doc] configuring plugin search directories
- Bug 1117293 - –installroot proceeds with an undefined conf.releasever
- Bug 1117678 - [py3] proper binary for python3-dnf
- Bug 1118178 - should create /etc/dnf/plugins directory
- Bug 1118796 - TypeError: must be unicode, not str
- Bug 1119032 - python3-dnf-0.5.3 conflicts with dnf-0.5.2

## 0.5.3 Release Notes

A set of bugfixes related to i18n and Unicode handling. There is a `-4/-6` switch and a corresponding *ip_resolve* configuration option (both known from Yum) to force DNS resolving of hosts to IPv4 or IPv6 addresses.

0.5.3 comes with several extensions and clarifications in the API: notably *Transaction* is introspectible now, *Query.filter* is more useful with new types of arguments and we've hopefully shed more light on how a client is expected to setup the configuration *substitutions*.

Finally, plugin authors can now use a new *resolved()* hook.

API changes in 0.5.3:

- extended description given for *dnf.Base.fill_sack()*
- dnf.Base.select_group() has been dropped as announced in *0.4.18 Release Notes*

API additions in 0.5.3:

- *dnf.conf.Conf.substitutions*
- *dnf.package.Package.arch*
- *dnf.package.Package.buildtime*
- *dnf.package.Package.epoch*
- *dnf.package.Package.installtime*
- *dnf.package.Package.name*
- *dnf.package.Package.release*
- *dnf.package.Package.sourcerpm*
- *dnf.package.Package.version*
- *dnf.Plugin.resolved()*
- *dnf.query.Query.filter()* accepts suffixes for its argument keys now which change the filter semantics.
- *dnf.rpm*
- *dnf.transaction.TransactionItem*
- *dnf.transaction.Transaction* is iterable now.

Bugs fixed in 0.5.3:

- Bug 1047049 - distroverpkg default should be the same in dnf as in yum
- Bug 1067156 - [api] introspecting transaction
- Bug 1093420 - Add to dnf.conf ip_resolve entry
- Bug 1104757 - [api] yumvars not set fail fedora metadata download
- Bug 1105009 - Please update the queries api documentation
- Bug 1110800 - [abrt] dnf: parser.py:59:varReplace:UnicodeDecodeError: 'ascii' codec can't decode byte 0xc3 in position 6: ordinal not in range(128)
- Bug 1111569 - make "Transaction couldn't start (no root?)" error more clear
- Bug 1111997 - [abrt] dnf: base.py:1894:_retrievePublicKey:TypeError: not all arguments converted during string formatting
- Bug 1112669 - commands from dnf-plugins-core not mentioned in man page
- Bug 1112704 - NameError: global name 'i18n' is not defined

## 0.5.2 Release Notes

This release brings autoremove command that removes any package that was originally installed as a dependency (e.g. had not been specified as an explicit argument to the install command) and is no longer needed.

Enforced verification of SSL connections can now be disabled with the *sslverify setting*.

We have been plagued with many crashes related to Unicode and encodings since the 0.5.0 release. These have been cleared out now.

There's more: improvement in startup time, extended globbing semantics for input arguments and better search relevance sorting.

Bugs fixed in 0.5.2:

- Bug 963345 - [rfe] autoremove
- Bug 1073457 - Auto-completion issue with sudo dnf install
- Bug 1076045 - [rfe] ssl support
- Bug 1083679 - dnf globbing does not really work
- Bug 1092006 - [abrt] dnf: __init__.py:1704:parse_extcmds:ValueError: need more than 0 values to unpack
- Bug 1092777 - [abrt] dnf: __init__.py:403:run:UnicodeDecodeError: 'ascii' codec can't decode byte 0xd0 in position 0: ordinal not in range(128)
- Bug 1093888 - [search] put exact matches first
- Bug 1094594 - dnf history rollback tracebacks
- Bug 1095580 - [kernel] kernel-models-extra cannot be installed because a newer one is already installed
- Bug 1095861 - UnicodeDecodeError: 'ascii' codec can't decode byte 0xe2 in position 21: ordinal not in range(128)
- Bug 1096506 - promoting 'install a' to 'install b' when b obsoletes a

## 0.5.1 Release Notes

Bugfix release with several internal cleanups. One outstanding change for CLI users is that DNF is a lot less verbose now during the dependency resolving phase.

Bugs fixed in 0.5.1:

- Bug 1065882 - [delta] delta rpm download size statistics
- Bug 1081753 - do not emit uncaught exceptions from automatic makecache runs
- Bug 1089864 - [api] group persistor is not saved when using group_* API

## 0.5.0 Release Notes

The biggest improvement in 0.5.0 is complete support for groups and environments, including internal database of installed groups independent of the actual packages (concept known as groups-as-objects from Yum). Upgrading groups is supported now with `group upgrade` too.

To force refreshing of metadata before an operation (even if the data is not expired yet), the refresh option has been added.

Internally, the CLI went through several changes to allow for better API accessibility like granular requesting of root permissions.

API has got many more extensions, focusing on better manipulation with comps and packages. There are new entries in *Changes in DNF CLI compared to Yum* and *DNF User's FAQ* too.

Several resource leaks (file descriptors, noncollectable Python objects) were found and fixed.

API changes in 0.5.0:

- it is now recommended that either *dnf.Base.close()* is used, or that *dnf.Base* instances are treated as a context manager.

API extensions in 0.5.0:

- *dnf.Base.add_remote_rpms()*
- *dnf.Base.close()*
- *dnf.Base.group_upgrade()*
- *dnf.Base.resolve()* optionally accepts *allow_erasing* arguments now.
- *dnf.Base.package_downgrade()*
- *dnf.Base.package_install()*
- *dnf.Base.package_upgrade()*
- dnf.cli.demand.DemandSheet
- *dnf.cli.Command.base*
- *dnf.cli.Command.cli*
- *dnf.cli.Command.summary*
- dnf.cli.Command.usage
- *dnf.cli.Command.configure()*
- *dnf.cli.Cli.demands*
- *dnf.comps.Package*
- *dnf.comps.Group.packages_iter()*
- *dnf.comps.MANDATORY* etc.

Bugs fixed in 0.5.0:

- Bug 1029022 - RFE: API for reverse group operations
- Bug 1051869 - [abrt] dnf: comps.py:62:_by_pattern:TypeError: expected string or buffer
- Bug 1061780 - Dnf allows installing conflicting packages
- Bug 1062884 - [api] Allow dnf plugin to set exit code
- Bug 1062889 - [RFE] [api] add class attribute to dnf.cli.Command which will say it is root-only plugin
- Bug 1063666 - [comps] [environments] dnf group remove “GNOME Desktop”
- Bug 1064211 - Dnf should provide logrotate script
- Bug 1064226 - [rfe] cmdline option forcing cache refresh
- Bug 1073859 - [api][rfe] access to dnf.Base.goal_parameters.allow_uninstall
- Bug 1076884 - [py3] highlight characters broken under py3
- Bug 1079519 - [api][rfe] install .rpm files with dnf API
- Bug 1079932 - [api][rfe] public api for getting all packages in a group
- Bug 1080331 - dnf group list fail to run with noroot plugin
- Bug 1080489 - [RFE] dnf exits with an error when at least one srpm is present on the command line

- Bug 1082230 - TypeError in dnf history info <id>
- Bug 1083432 - group is marked as installed, even if install of the packages failed
- Bug 1083767 - dnf documents cached packages in a way which does not agree with reality
- Bug 1084139 - dnf manpage "manipualte" typos
- Bug 1084553 - [doc] dnf docs describe '–best' option in undecipherabke manner
- Bug 1088166 - [doc] DNF doesn't say you need root rights to run dnf update

## 0.4.19 Release Notes

Arriving one week after 0.4.18, the 0.4.19 mainly provides a fix to a traceback in group operations under non-root users.

DNF starts to ship separate translation files (.mo) starting with this release.

Bugs fixed in 0.4.19:

- Bug 1077173 - IOError: [Errno 13] Permission denied: '/var/lib/dnf/groups.json'
- Bug 1078832 - TypeError: '_DownloadErrors' object is not iterable
- Bug 1079621 - Incorrect package count when some packages already downloaded

## 0.4.18 Release Notes

Support for `dnf distro-sync <spec>` finally arrives in this version.

DNF has moved to handling groups as objects, tagged installed/uninstalled independently from the actual installed packages. This has been in Yum as the `group_command=objects` setting and the default in recent Fedora releases. There are API extensions related to this change as well as two new CLI commands: `group mark install` and `group mark remove`.

API items deprecated in 0.4.8 and 0.4.9 have been dropped in 0.4.18, in accordance with our deprecating-label.

API changes in 0.4.18:

- `dnf.queries` has been dropped as announced in *0.4.8 Release Notes*
- `dnf.exceptions.PackageNotFoundError` has been dropped from API as announced in *0.4.9 Release Notes*
- *`dnf.Base.install()`* no longer has to return the number of marked packages as announced in *0.4.9 Release Notes*

API deprecations in 0.4.18:

- `dnf.Base.select_group()` is deprecated now. Please use *`group_install()`* instead.

API additions in 0.4.18:

- *`dnf.Base.group_install()`*
- *`dnf.Base.group_remove()`*

Bugs fixed in 0.4.18:

- Bug 963710 - RFE: allow package specs for distro-sync
- Bug 1067136 - [comps] no way to install optional packages from a group

- Bug 1071212 - dnf doesn't skip unreadable repository files
- Bug 1071501 - [drpm] fallback to full RPM download on drpm failure

## 0.4.17 Release Notes

This release fixes many bugs in the downloads/DRPM CLI area. A bug got fixed preventing a regular user from running read-only operations using `--cacheonly`. Another fix ensures that `metadata_expire=never` setting is respected. Lastly, the release provides three requested API calls in the repo management area.

API additions in 0.4.17:

- *dnf.repodict.RepoDict.all()*
- *dnf.repodict.RepoDict.get_matching()*
- *dnf.repo.Repo.set_progress_bar()*

Bugs fixed in 0.4.17:

- Bug 1059704 - [api] dnf.base.Base.read_comps() fails when there's no comps
- Bug 1058224 - the system cache should be world readable by default
- Bug 1069538 - DNF doesn't support "never" or "-1" for metadata_expire option unlike yum
- Bug 1070598 - problems downloading large files (and counting past 100%)
- Bug 1070710 - dnf.Base.read_comps() fails in Python3, works ok in Python2
- Bug 1071323 - [api] [rfe]: dnf.repo & dnf.repodict API extention
- Bug 1071455 - [i18n] Wrong characters in Japanese translatated message
- Bug 1071501 - [drpm] fallback to full RPM download on drpm failure
- Bug 1071518 - curl error
- Bug 1071677 - [doc] faq suggests it's a good idea to install rawhide packages on stable releases

## 0.4.16 Release Notes

The refactorings from 0.4.15 are introducing breakage causing the background `dnf makecache` runs traceback. This release fixes that.

Bugs fixed in 0.4.16:

- Bug 1069996 - [abrt] dnf: repo.py:312:start:AttributeError: 'NoneType' object has no attribute 'start'

## 0.4.15 Release Notes

Massive refactoring of the downloads handling to provide better API for reporting download progress and fixed bugs are the main things brought in 0.4.15.

API additions in 0.4.15:

- *dnf.exceptions.DownloadError*

- *dnf.Base.download_packages()* now takes the optional *progress* parameter and can raise *DownloadError*.
- *dnf.callback.Payload*
- *dnf.callback.DownloadProgress*
- *dnf.query.Query.filter()* now also recognizes `provides` as a filter name.

Bugs fixed in 0.4.15:

- Bug 1048788 - [abrt] dnf: __init__.py:183:install:ArchException: Used arch is unknown.
- Bug 1065728 - [doc] Metadata expire configurable per repository
- Bug 1065879 - [API] provides is missing in queries documentation
- Bug 1065959 - dnf tab completion should use "distro-sync" instead of "distribution-synchronization"
- Bug 1066743 - [doc] 'dnf clean' should function or warn when run as normal user

## 0.4.14 Release Notes

This quickly follows 0.4.13 to address the issue of crashes when DNF output is piped into another program.

API additions in 0.4.14:

- *Repo.pkgdir*

Bugs fixed in 0.4.14:

- Bug 1062390 - dnf.rpmUtils.arch.getBaseArch() doesn't return identical results to yum's rpmUtils.arch.getBaseArch()
- Bug 1062847 - dnf man page errors
- Bug 1063022 - [abrt] dnf: i18n.py:38:write:UnicodeEncodeError: 'ascii' codec can't encode character u'\xe9' in position 12: ordinal not in range(128)
- Bug 1064148 - DNF attempts to remove packages from local file-based repository

## 0.4.13 Release Notes

0.4.13 finally ships support for delta RPMS. Enabling this can save some bandwidth (and use some CPU time) when downloading packages for updates.

Support for bash completion is also included in this version. It is recommended to use the `generate_completion_cache` plugin to have the completion work fast. This plugin will be also shipped with `dnf-plugins-core-0.0.3`.

The *keepcache* config option has been readded.

Bugs fixed in 0.4.13:

- Bug 909468 - [rfe] delta rpm support
- Bug 1030440 - [RFE] bash completion support
- Bug 1046244 - dnf doesn't keep the downloaded packages in the cache
- Bug 1055051 - things like ' dnf install "kernel-devel-uname-r = 3.12.8-300.fc20.x86_64" ' don't work
- Bug 1056400 - DNF not support url installation for a package

## 0.4.12 Release Notes

This release disables fastestmirror by default as we received many complains about it. There are also several bugfixes, most importantly an issue has been fixed that caused packages installed by Anaconda be removed together with a depending package. It is now possible to use `bandwidth` and `throttle` config values too.

Bugs fixed in 0.4.12:

- Bug 1045737 - [RFE] 'throttle' option.
- Bug 1048468 - dnf –cacheonly list installed tries to download repodata
- Bug 1048488 - AssertionError on invalid error log level
- Bug 1049025 - Incorrect dependency check on removal
- Bug 1051554 - disable fastestmirror by default (and make it configurable)

## 0.4.11 Release Notes

This is mostly a bugfix release following quickly after 0.4.10, with many updates to documentation.

API additions in 0.4.11:

- *`Plugin.read_config()`*
- *`repo.Metadata`*
- *`repo.Repo.metadata`*

API changes in 0.4.11:

- *`Conf.pluginpath`* is no longer hard coded but depends on the major Python version.

Bugs fixed in 0.4.11:

- Bug 1048402 - [abrt] dnf: cli.py:1521:search:AttributeError: 'module' object has no attribute 'exception'
- Bug 1048572 - dnf resolve failure
- Bug 1048716 - man page does not mention sub command remove
- Bug 1048719 - Minor error in 'dnf –help', using the package name Yum instead of dnf
- Bug 1048988 - [plugins] [api] add a hook for MD refresh

## 0.4.10 Release Notes

0.4.10 is a bugfix release that also adds some long-requested CLI features and extends the plugin support with two new plugin hooks. An important feature for plugin developers is going to be the possibility to register plugin's own CLI command, available from this version.

`dnf history` now recognizes `last` as a special argument, just like other history commands.

`dnf install` now accepts group specifications via the `@` character.

Support for the `--setopt` option has been readded from Yum.

API additions in 0.4.10:

- *Command Line Interface Hooks*

- *Plugin.name*
- *Plugin.__init__()* now specifies the second parameter as an instance of *.cli.Cli*
- *Plugin.sack()*
- *Plugin.transaction()*
- *repo.repo_id_invalid()*

API changes in 0.4.10:

- Plugin authors must specify *Plugin.name* when authoring a plugin.

Bugs fixed in 0.4.10:

- Bug 967264 - [plugins] Support post-transaction actions
- Bug 1018284 - [rfe] [api] refactor check for valid id into separate function
- Bug 1035164 - [abrt] dnf-0.4.7-1.fc20: miscutils.py:61:checkSig:OSError: [Errno 2] Adresář nebo soubor neexistuje: '/var/cache/dnf/x86_64/20/updates-testing/packages/kernel-3.11.9-300.fc20.x86_64.rpm'
- Bug 1036147 - [locking] Traceback for dnf search if running under su - USER
- Bug 1036211 - Install groups using @ sign
- Bug 1038403 - [abrt] dnf-0.4.9-1.fc20: base.py:1832:handle_erase:AssertionError
- Bug 1038937 - [plugins] [api] Add hook for metadata download complete event
- Bug 1040255 - dnf show wrong Installed size when install packages
- Bug 1044502 - [abrt] dnf: base.py:125:_add_repo_to_sack:UnicodeDecodeError: 'ascii' codec can't decode byte 0xd0 in position 113: ordinal not in range(128)
- Bug 1044981 - [doc] [rfe] support setopt
- Bug 1044999 - [doc] show dependencies during update

## 0.4.9 Release Notes

Several Yum features are revived in this release. `dnf history rollback` now works again. The `history userinstalled` has been added, it displays a list of ackages that the user manually selected for installation on an installed system and does not include those packages that got installed as dependencies.

We're happy to announce that the API in 0.4.9 has been extended to finally support plugins. There is a limited set of plugin hooks now, we will carefully add new ones in the following releases. New marking operations have ben added to the API and also some configuration options.

An alternative to `yum shell` is provided now for its most common use case: replacing a non-leaf package with a conflicting package is achieved by using the `--allowerasing` switch now.

API additions in 0.4.9:

- *Plugin Interface*
- *Logging*
- *Base.read_all_repos()*
- *Base.reset()*
- *Base.downgrade()*
- *Base.remove()*

- *Base.upgrade()*
- *Base.upgrade_all()*
- *Conf.pluginpath*
- *Conf.reposdir*

API deprecations in 0.4.9:

- PackageNotFoundError is deprecated for public use. Please catch *MarkingError* instead.
- It is deprecated to use *Base.install()* return value for anything. The method either returns or raises an exception.

Bugs fixed in 0.4.9:

- Bug 884615 - RFE: Provide list of packages installed on top of the system
- Bug 963137 - [rfe] perform complex packaging operations as a CLI user
- Bug 991038 - lacks history rollback
- Bug 1032455 - [abrt] dnf-0.4.7-1.fc20: decoder.py:383:raw_decode:ValueError: No JSON object could be decoded
- Bug 1034607 - dnf does not work with media repo
- Bug 1036116 - dnf install failed with exception

## 0.4.8 Release Notes

There are mainly internal changes, new API functions and bugfixes in this release.

Python 3 is fully supported now, the Fedora builds include the Py3 variant. The DNF program still runs under Python 2.7 but the extension authors can now choose what Python they prefer to use.

This is the first version of DNF that deprecates some of its API. Clients using deprecated code will see a message emitted to stderr using the standard Python warnings module. You can filter out *dnf.exceptions.DeprecationWarning* to suppress them.

API additions in 0.4.8:

- *dnf.Base.sack*
- *dnf.conf.Conf.cachedir*
- *dnf.conf.Conf.config_file_path*
- *dnf.conf.Conf.persistdir*
- *dnf.conf.Conf.read()*
- *dnf.package.Package*
- *dnf.query.Query*
- *dnf.subject.Subject*
- *dnf.repo.Repo.__init__()*
- *dnf.sack.Sack*
- *dnf.selector.Selector*
- *dnf.transaction.Transaction*

API deprecations in 0.4.8:

- `dnf.queries` is deprecated now. If you need to create instances of *`Subject`*, import it from *`dnf.subject`*. To create *`Query`* instances it is recommended to use *`sack.query()`*.

Bugs fixed in 0.4.8:

- Bug 1014563 - Support Python 3 in DNF
- Bug 1029948 - dnf does not show all desktop environments in dnf groups list output
- Bug 1030998 - install doesn't work for '*/binaryname' patterns
- Bug 1030297 - dnf doesn't support http urls
- Bug 1030980 - dnf downgrade PACKAGE, misleading message about possible typo if PACKAGE is not installed

## 0.4.7 Release Notes

We start to publish the *DNF API Reference* with this release. It is largely incomprehensive at the moment, yet outlines the shape of the documentation and the process the project is going to use to maintain it.

There are two Yum configuration options that were dropped: *group_package_types* and *upgrade_requirements_on_install*.

Bugs fixed in 0.4.7:

- Bug 1019170 - handling wildcards in the version
- Bug 1024776 - librepo version mismatch (no attribute 'LRO_FASTESTMIRRORCACHE')
- Bug 1025650 - [abrt] dnf-0.4.6-1.fc20: i18n.py:119:ucd:UnicodeEncodeError: 'ascii' codec can't encode character u'\xe9' in position 51: ordinal not in range(128)

## 0.4.6 Release Notes

0.4.6 brings two new major features. Firstly, it is the revival of `history undo`, so transactions can be reverted now. Secondly, DNF will now limit the number of installed kernels and *installonly* packages in general to the number specified by *installonly_limit* configuration option.

DNF now supports the `group summary` command and one-word group commands no longer cause tracebacks, e.g. `dnf grouplist`.

There are vast internal changes to `dnf.cli`, the subpackage that provides CLI to DNF. In particular, it is now better separated from the core.

The hawkey library used against DNF from with this versions uses a recent RPMDB loading optimization in libsolv that shortens DNF startup by seconds when the cached RPMDB is invalid.

We have also added further fixes to support Python 3 and enabled librepo's fastestmirror caching optimization to tighten the download times even more.

Bugs fixed in 0.4.6:

- Bug 878348 - 'dnf history undo' support missing
- Bug 880524 - [kernel] managing multiple kernel packages
- Bug 1019957 - 'dnf groups summary' traceback (global name 'patterns' is not defined)
- Bug 1020101 - 'dnf grouplist' throws NameError

- Bug 1020934 - traceback in "dnf repolist"
- Bug 1023486 - groups operations failing due to a libcomps API change in its latest version (KeyError: 'en_US.UTF-8')

## 0.4.5 Release Notes

A serious bug causing tracebacks during package downloads made it into 0.4.4 and this release contains a fix for that. Also, a basic proxy support has been readded now.

Bugs fixed in 0.4.5:

- Bug 1021087 - traceback during download after a transaction with obsoletes

## 0.4.4 Release Notes

The initial support for Python 3 in DNF has been merged in this version. In practice one can not yet run the `dnf` command in Py3 but the unit tests already pass there. We expect to give Py3 and DNF heavy testing during the Fedora 21 development cycle and eventually switch to it as the default. The plan is to drop Python 2 support as soon as Anaconda is running in Python 3.

Minor adjustments to allow Anaconda support also happened during the last week, as well as a fix to a possibly severe bug that one is however not really likely to see with non-devel Fedora repos:

- Bug 1017278 - dnf does not notice file conflicts

## 0.4.3 Release Notes

This is an early release to get the latest DNF out with the latest librepo fixing the Too many open files bug.

In Fedora, the spec file has been updated to no longer depend on precise versions of the libraries so in the future they can be released independently.

This release sees the finished refactoring in error handling during basic operations and adds support for `group remove` and `group info` commands, i.e. the following two bugs:

- Bug 1013764 - DNF group remove doesn't work
- Bug 1013773 - DNF group info doesn't work

## 0.4.2 Release Notes

DNF now downloads packages for the transaction in parallel with progress bars updated to effectively represent this. Since so many things in the downloading code were changing, we figured it was a good idea to finally drop urlgrabber dependency at the same time. Indeed, this is the first version that doesn't require urlgrabber for neither build nor run.

Similarly, since librepo started to support this, downloads in DNF now use the fastests mirrors available by default.

The option to *specify repositories' costs* has been readded.

Internally, DNF has seen first part of ongoing refactorings of the basic operations (install, update) as well as a couple of new API methods supporting development of extensions.

These bugzillas are fixed in 0.4.2:

- Bug 909744 - [rfe] support parallel downloads
- Bug 984529 - [rfe] fastest mirror
- Bug 967798 - [RFE] repositories costs
- Bug 995459 - conflicting requests when trying to install locally available packages

## 0.4.1 Release Notes

The focus of this release was to support our efforts in implementing the DNF Payload for Anaconda, with changes on the API side of things (better logging, new `Base.reset()` method).

Support for some irrelevant config options has been dropped (`kernelpkgnames`, `exactarch`, `rpm_check_debug`). We also no longer detect metalinks in the `mirrorlist` option (see Fedora bug 948788).

DNF is on its way to drop the urlgrabber dependency and the first set of patches towards this goal is already in.

Expect the following bugs to go away with upgrade to 0.4.1:

- Bug 998859 - "dnf upgrade –best" will always install 64bits package when 32bit package is installed
- Bug 1006366 - dnf makecache runs on LiveCD
- Bug 1008444 - confusing message when removing packages
- Bug 1003220 - [abrt] dnf-0.3.11-1.git7d717c7.fc19: output.py:838:_enc:TypeError: object of type 'NoneType' has no len()

## 0.4.0 Release Notes

The new minor version brings many internal changes to the comps code, most comps parsing and processing is now delegated to libcomps by Jindřich Luža.

The `overwrite_groups` config option has been dropped in this version and DNF acts if it was 0, that is groups with the same name are merged together.

The currently supported groups commands (`group list` and `group install`) are documented on the manpage now.

The 0.4.0 version is the first one supported by the DNF Payload for Anaconda and many changes since 0.3.11 make that possible by cleaning up the API and making it more sane (cleanup of `yumvars` initialization API, unifying the RPM transaction callback objects hierarchy, slimming down `dnf.rpmUtils.arch`, improved logging).

Fixes for the following are contained in this version:

- Bug 997403 - Error message instructs users to invoke 'yum' rather than 'dnf' to fix a problem
- Bug 1002508 - Updated to install any packages "An Curl handle error: Unsupported protocol"
- Bug 1002798 - Error when checking librepo explicit requires

## 0.3.11 Release Notes

The default multilib policy configuration value is `best` now. This does not pose any change for the Fedora users because exactly the same default had been previously achieved by a setting in `/etc/dnf/dnf.conf` shipped with the Fedora package.

An important fix to the repo module speeds up package downloads again is present in this release. The full list of fixes is:

- Bug 979042 - Slow download of packages
- Bug 977753 - Repository "@System" instead of "fedora"
- Bug 996138 - [abrt] install_grouplist:UnicodeDecodeError during 'dnf group install'
- Bug 993916 - 'upgrade' command displays old version numbers in summary, rather than new version numbers

## 0.3.10 Release Notes

The only major change is that `skip_if_unavailable` is *enabled by default now*.

A minor release otherwise, mainly to get a new version of DNF out that uses a fresh librepo. The following issues are now a thing of the past:

- Bug 977661 - [abrt] Ctrl+C ends up with a TypeError traceback involving Base.__exit__() and Repo._handle_load()
- Bug 984483 - default to skip_if_unavailable=True
- Bug 986545 - dnf mismatching it's librepo dep in rawhide

## 0.3.9 Release Notes

This is a quick bugfix release dealing with reported bugs and tracebacks:

- Bug 964584 - dnf doesn't replace $releasever/$basearch in mirror url
- Bug 979942 - [abrt] cli.py:1015:_configure_repos:KeyError: <repo id>
- Bug 980227 - [locking] OSError when a long running transaction gets metadata files written over by a concurrent 'dnf makecache'
- Bug 981310 - NameError: global name 'to_utf8' is not defined

## 0.3.8 Release Notes

A new locking module has been integrated in this version, clients should see the message about DNF lock being taken less often.

Panu Matilainen has submitted many patches to this release to cleanup the RPM interfacing modules.

The following bugs are fixed in this release:

- Bug 908491 - [rfe] list recent
- Bug 968159 - repo_rpmmd.c: base:xml attr in <location> is ignored
- Bug 974427 - dnf fails on all commands with fatal error: "global name 'os' is not defined"
- Bug 974866 - dnf crashes if neither baseurl nor mirrorlist are defined
- Bug 976652 - [abrt] UnicodeEncodeError with 'dnf info checkpolicy'
- Bug 975858 - dnf can't install in an empty installroot

## 0.3.7 Release Notes

This is a bugfix release:

- Bug 916662 - dnf remove wants to install new packages
- Bug 967732 - [abrt] dnf-0.3.6-1.git24ce938.fc19: __init__.py:364:hashopen:DBNoSuchFileError: (2, 'Datei oder Verzeichnis nicht gefunden')

## 0.3.6 Release Notes

This is a bugfix release, including the following fixes:

- Bug 966372 - dnf prints an empty python list after running without root privileges
- Bug 965410 - "dnf clean expire-cache" does not have any effect
- Bug 963627 - dnf shows way over 100% in its progress bar
- Bug 965114 - When running dnf using file:/// as the media, it tries to erase packages in file:///
- Bug 964467 - group install is broken
- Bug 963680 - dnf dies with a traceback during update
- Bug 963133 - name is misprinted when package is not found

## 0.3.5 Release Notes

Besides few fixed bugs this version should not present any differences for the user. On the inside, the transaction managing mechanisms have changed drastically, bringing code simplification, better maintainability and better testability.

In Fedora, there is a change in the spec file effectively preventing the makecache timer from running *immediately after installation*. The timer service is still enabled by default, but unless the user starts it manually with `systemctl start dnf-makecache.timer` it will not run until after the first reboot. This is in alignment with Fedora packaging best practices.

The following bugfixes are included in 0.3.5:

- Bug 958452 - dnf update Produces Log Messages Involving "ts_done state"
- Bug 959990 - dnf dist-sync recommended in manual page but is missing from dnf's command line options
- Bug 961549 - [abrt] dnf-0.3.4-1.git03fd687.fc19: __init__.py:826:perform:Exception: (10, 'Error HTTP/FTP status code: 404', 404)
- Bug 962188 - obsoletes sometimes result in "ERROR with transaction check vs depsolve"

## 0.3.4 Release Notes

0.3.4 is the first DNF version since the fork from Yum that is able to manipulate the comps data. In practice, `dnf group install <group name>` works again. No other group commands are supported yet.

Support for `librepo-0.0.4` and related cleanups and extensions this new version allows are included (see the buglist below)

This version has also improved reporting of obsoleted packages in the CLI (the Yum-style "replacing <package-nevra>" appears in the textual transaction overview).

The following bugfixes are included in 0.3.4:

- Bug 887317 - Dnf should beter report upgrades via obsoletes.
- Bug 914919 - DNF does not have group install
- Bug 922667 - the cronned 'dnf makecache' runs should give up quickly if there's network/checksumming issues

## 0.3.3 Release Notes

The improvements in 0.3.3 are only API changes to the logging. There is a new module `dnf.logging` that defines simplified logging structure compared to Yum, with fewer logging levels and simpler usage for the developers. The RPM transaction logs are no longer in `/var/log/dnf.transaction.log` but in `/var/log/dnf.rpm.log` by default.

The exception classes were simplified and moved to `dnf.exceptions`.

The following bugs are fixed in 0.3.3:

- Bug 950722 - [abrt] dnf-0.3.2-1.gitf3818b4.fc20: queries.py:179:_construct_result:AttributeError: 'NoneType' object has no attribute 'query'
- Bug 903775 - Update command is deprecated but nomenclature still being used

## 0.3.2 Release Notes

The major improvement in this version is in speeding up syncing of repositories using metalink by looking at the repomd.xml checksums. This effectively lets DNF cheaply refresh expired repositories in cases where the original has not changed: for instance the main Fedora repository is refreshed with one 30 kB HTTP download. This functionality is present in the current Yum but hasn't worked in DNF since 3.0.0.

Otherwise this is mainly a release fixing bugs and tracebacks. The following reported bugs are fixed:

- Bug 947258 - dnf doesn't erase excluded packages
- Bug 889202 - [rfe] [repos] add support for skip_if_unavailable in repository config
- Bug 923384 - DNF uses yum web signature

## 0.3.1 Release Notes

0.3.1 brings mainly changes to the automatic metadata synchronization. In Fedora, `dnf makecache` is triggered via SystemD timers now and takes an optional `background` extra-argument to run in resource-considerate mode (no syncing when running on laptop battery, only actually performing the check at most once every three hours). Also, the IO and CPU priorities of the timer-triggered process are lowered now and shouldn't as noticeably impact the system's performance.

The administrator can also easily disable the automatic metadata updates by setting *metadata_timer_sync* to 0.

The default value of *metadata_expire* was increased from 6 hours to 48 hours. In Fedora, the repos usually set this explicitly so this change is not going to cause much impact.

The following reported issues are fixed in this release:

- Bug 916657 - dnf remove wants to downgrade packages; fails on transaction check
- Bug 921294 - dnf downgrade failure
- Bug 922521 - DNF crashes when passing packages to it via pipe
- Bug 926871 - [abrt] download_packages:AttributeError: 'Repo' object has no attribute 'cache'
- Bug 878826 - [rfe] dnf could use systemd timer unit rather than crontabs
- Bug 922664 - an option for disabling the regular metadata sync
- Bug 892064 - hourly dnf makecache too frequent, also consider using ionice
- Bug 919769 - dnf runs makecache hourly despite laptop power state

CHAPTER 8

# Changes in DNF CLI compared to Yum

## `--skip-broken`

For install command:

The `--skip-broken` option is alias for `--setopt=strict=0`. The both options could be used with DNF to skip all unavailable packages or packages with broken dependencies given to DNF command without raising the error causing the whole operation to fail. This behavior can be set as default in dnf.conf file. See *strict conf option*.

For upgrade command:

The semantics that was supposed to trigger in Yum with `--skip-broken` is now set for plain `dnf update` as a default. There is no need to use `--skip-broken` with `dnf upgrade` command. To use only the latest versions of packages in transactions, there is the `--best` command line switch.

## Update and Upgrade Commands are the Same

Invoking `dnf update` or `dnf upgrade`, in all their forms, has the same effect in DNF, with the latter being preferred. In Yum `yum upgrade` was exactly like `yum --obsoletes update`.

## `clean_requirements_on_remove` on by default

The *clean_requirements_on_remove* switch is on by default in DNF. It can thus be confusing to compare the "remove" operation results between DNF and Yum as by default DNF is often going to remove more packages.

## No `resolvedep` command

The Yum version of this command is maintained for legacy reasons only. The user can just do `dnf provides` to find out what package gives a particular provide.

## No `deplist` command

Alternative to Yum `deplist` command to find out dependencies of the package is `dnf repoquery --deplist` using *repoquery command*.

**Note:** Alternatively there is a YUM compatibility support where `yum deplist` is alias for `dnf repoquery --deplist` command

## Excludes and repo excludes apply to all operations

Yum only respects excludes during installs and upgrades. DNF extends this to all operations, among others erasing and listing. If you e.g. want to see a list of all installed `python-f*` packages but not any of the Flask packages, the following will work:

```
dnf -x '*flask*' list installed 'python-f*'
```

## Yum's conf directive `includepkgs` is just `include`

`include` directive name of [main] and Repo configuration is more logical and better named counterpart of `exclude` in DNF.

## `dnf remove kernel` deletes all packages called `kernel`

In Yum, the running kernel is spared. There is no reason to keep this in DNF, the user can always specify concrete versions on the command line, e.g.:

```
dnf remove kernel-3.9.4
```

## `dnf provides /bin/<file>` does not find any packages on Fedora

After UsrMove there's no directory `/bin` on Fedora systems and no files get installed there, `/bin` is only a symlink created by the `filesystem` package to point to `/usr/bin`. Resolving the symlinks to their real path would only give the user false sense that this works while in fact provides requests using globs such as:

```
dnf provides /b*/<file>
```

will fail still (as it does in Yum now). To find what provides a particular binary use the actual path for binaries on Fedora:

```
dnf provides /usr/bin/<file>
```

Also see related Fedora bugzillas 982947 and 982664.

## `skip_if_unavailable` enabled by default

The important system repos should never be down and we see the third party repos down often enough to warrant this change. Note that without this setting and without an explicit `skip_if_unavailable=True` in the relevant repo .ini file Yum immediately stops on a repo error, confusing and bothering the user.

See the related Fedora bug 984483.

## `overwrite_groups` dropped, comps functions acting as if always disabled

This config option has been dropped. When DNF sees several groups with the same group id it merges the groups' contents together.

## `mirrorlist_expire` dropped

To simplify things for the user, DNF uses `metadata_expire` for both expiring metadata and the mirrorlist file (which is a kind of metadata itself).

## metalink not recognized in the `mirrorlist` repo option

The following part of `yum.conf(5)` no longer applies for the `mirrorlist` option:

> As a special hack is the mirrorlist URL contains the word "metalink" then the value of mirrorlist is copied to metalink (if metalink is not set).

The relevant repository configuration files have been fixed to respect this, see the related Fedora bug 948788.

## `alwaysprompt` dropped

Unsupported to simplify the configuration.

## `group_package_types` dropped

Done to simplify the configuration. User will typically want to decide what packages to install per-group and not via a global setting:

```
dnf group install with-optional Editors
```

## `upgrade_requirements_on_install` dropped

Dropping this config option with blurry semantics simplifies the configuration. DNF behaves as if this was disabled. If the user wanted to upgrade everything to the latest version she'd simply use `dnf upgrade`.

## `dnf history rollback` check dropped

DNF tolerates the use of other package managers. Then it is possible that not all changes to RPMDB are stored in the history of transactions. Therefore, DNF does not fail if such a situation is encountered and thus the `force` option is not needed anymore.

## Packages replacement without `yum swap`

Time after time one needs to remove an installed package and replace it with a different one, providing the same capabilities while other packages depending on these capabilities stay installed. Without (transiently) breaking consistency of the package database this can be done by performing the remove and the install in one transaction. The common way to setup such transaction in DNF is to use `dnf shell` or use `--allowerasing` switch.

E.g. say you want to replace `A` (providing `P`) with B (also providing `P`, conflicting with `A`) without deleting `C` (which requires `P`) in the process. Use:

```
dnf --allowerasing install B
```

This command is equal to `yum swap A B`.

DNF provides swap command but only `dnf swap A B` syntax is supported

## Dependency processing details are not shown in the CLI

During its depsolving phase, Yum outputs lines similar to:

```
---> Package rubygem-rhc.noarch 0:1.16.9-1.fc19 will be an update
--> Processing Dependency: rubygem-net-ssh-multi >= 1.2.0 for package: rubygem-rhc-1.
↪16.9-1.fc19.noarch
```

DNF does not output information like this. The technical reason is that depsolver below DNF always considers all dependencies for update candidates and the output would be very long. Secondly, even in Yum this output gets confusing very quickly especially for large transactions and so does more harm than good.

See the the related Fedora bug 1044999.

## `dnf provides` complies with the Yum documentation of the command

When one executes:

```
yum provides sandbox
```

Yum applies extra heuristics to determine what the user meant by `sandbox`, for instance it sequentially prepends entries from the `PATH` environment variable to it to see if it matches a file provided by some package. This is an undocumented behavior that DNF does not emulate. Just typically use:

```
dnf provides /usr/bin/sandbox
```

or even:

```
dnf provides '*/sandbox'
```

to obtain similar results.

## `--enableplugin` not recognized

This switch has been dropped. It is not documented for Yum and of a questionable use (all plugins are enabled by default).

## Bandwidth limiting

DNF supports the `throttle` and `bandwidth` options familiar from Yum. Contrary to Yum, when multiple downloads run simultaneously the total downloading speed is throttled. This was not possible in Yum since downloaders ran in different processes.

## `installonlypkgs` config option

Compared to Yum, DNF appends list values from the `installonlypkgs` config option to DNF defaults, where YUM overwrites the defaults by option values.

## The usage of Delta RPM files

The boolean `deltarpm` option controls whether delta RPM files are used. Compared to Yum, DNF does not support `deltarpm_percentage` and instead chooses some optimal value of DRPM/RPM ratio to decide whether using deltarpm makes sense in the given case.

## Handling .srpm files and non-existent packages

DNF will terminate early with an error if a command is executed requesting an installing operation on a local `.srpm` file:

```
$ dnf install fdn-0.4.17-1.fc20.src.rpm tour-4-6.noarch.rpm
Error: Will not install a source rpm package (fdn-0.4.17-1.fc20.src).
```

The same applies for package specifications that does not match any available package.

Yum will only issue warning in this case and continue installing the "tour" package. The rationale behind the result in DNF is that a program should terminate with an error if it can not fulfill the CLI command in its entirety.

## Promoting package to install to a package that obsoletes it

DNF will not magically replace a request for installing package `X` to installing package `Y` if `Y` obsoletes `X`. Yum does this if its `obsoletes` config option is enabled but the behavior is not properly documented and can be harmful.

See the the related Fedora bug 1096506 and guidelines for renaming and obsoleting packages in Fedora.

## Behavior of `--installroot` option

DNF offer more predictable behavior of installroot. DNF differently handles path from `--config` command-line option, where this path is always related to host system (Yum combines this path with installroot). Reposdir is also slightly differently handled, if one path of reposdirs exists inside of installroot, than repos are strictly taken from installroot (Yum tests each path from reposdir separately and use installroot path if existed). See detailed description for *--installroot* option.

# CHAPTER 9

## Changes in DNF plugins compared to Yum plugins

| Original Yum tool | DNF command/option | Package |
|---|---|---|
| `yum check` | *dnf repoquery* `--unsatisfied` | `dnf` |
| `yum-langpacks` | | `dnf-langpacks` |
| `yum-plugin-auto-update-debuginfo` | option in `debuginfo-install.conf` | `dnf-plugins-core` |
| `yum-plugin-copr` | dnf copr | `dnf-plugins-core` |
| `yum-plugin-fastestmirror` | `fastestmirror` option in dnf.conf | `dnf` |
| `yum-plugin-fs-snapshot` | | `dnf-plugins-extras-snapper` |
| `yum-plugin-local` | | `dnf-plugins-extras-local` |
| `yum-plugin-merge-conf` | | `dnf-plugins-extras-rpmconf` |
| `yum-plugin-priorities` | `priority` option in dnf.conf | `dnf` |
| `yum-plugin-remove-with-leaves` | `dnf autoremove` | `dnf` |
| `yum-plugin-show-leaves` | | `dnf-plugins-extras-show-leaves` |
| `yum-plugin-versionlock` | | `dnf-plugins-extras-versionlock` |

Plugins that have not been ported yet:

`yum-plugin-aliases`, `yum-plugin-changelog`, `yum-plugin-filter-data`, `yum-plugin-keys`, `yum-plugin-list-data`, `yum-plugin-post-transaction-actions`, `yum-plugin-protectbase`, `yum-plugin-ps`, `yum-plugin-puppetverify`, `yum-plugin-refresh-updatesd`, `yum-plugin-rpm-warm-cache`, `yum-plugin-tmprepo`, `yum-plugin-tsflags`, `yum-plugin-upgrade-helper`, `yum-plugin-verify`

Feel free to file a RFE for missing functionality if you need it.

# CHAPTER 10

# Changes in DNF plugins compared to Yum utilities

All ported yum tools are now implemented as DNF plugins.

| Original Yum tool | New DNF command | Package |
|---|---|---|
| `debuginfo-install` | dnf debuginfo-install | `dnf-plugins-core` |
| `find-repos-of-install` | dnf list installed | `dnf` |
| `needs-restarting` | dnf tracer | `dnf-plugins-extras-tracer` |
| `package-cleanup` | *dnf list*, *dnf repoquery* | `dnf`, `dnf-plugins-core` |
| `repoclosure` | dnf repoclosure | `dnf-plugins-extras-repoclosure` |
| `repo-graph` | dnf repograph | `dnf-plugins-extras-repograph` |
| `repomanage` | dnf repomanage | `dnf-plugins-extras-repomanage` |
| `repoquery` | *dnf repoquery* | `dnf` |
| `reposync` | dnf reposync | `dnf-plugins-core` |
| `repotrack` | dnf download | `dnf-plugins-core` |
| `yum-builddep` | dnf builddep | `dnf-plugins-core` |
| `yum-config-manager` | dnf config-manager | `dnf-plugins-core` |
| `yum-debug-dump` | dnf debug-dump | `dnf-plugins-extras-debug` |
| `yum-debug-restore` | dnf debug-restore | `dnf-plugins-extras-debug` |
| `yumdownloader` | dnf download | `dnf-plugins-core` |

Detailed table for `package-cleanup` replacement:

| | |
|---|---|
| `package-cleanup --dupes` | `dnf repoquery --duplicates` |
| `package-cleanup --leaves` | `dnf repoquery --unneeded` |
| `package-cleanup --orphans` | `dnf repoquery --extras` |
| `package-cleanup --oldkernels` | `dnf repoquery --installonly` |
| `package-cleanup --problems` | `dnf repoquery --unsatisfied` |
| `package-cleanup --cleandupes` | `dnf remove --duplicates` |
| `package-cleanup --oldkernels` | `dnf remove --oldinstallonly` |

Utilities that have not been ported yet:

`repodiff`, `repo-rss`, `show-changed-rco`, `show-installed`, `verifytree`, `yum-groups-manager`

Take a look at FAQ about yum to DNF migration. Feel free to file a RFE for missing functionality if you need it.

CHAPTER 11

# Changes in DNF hook api compared to Yum

This table provides what alternative hooks are available in DNF compared to yum.

| Hook Number | Yum hook | DNF hook |
|---|---|---|
| 1 | config | init |
| 2 | postconfig | init |
| 3 | init | init |
| 4 | predownload | |
| 5 | postdownload | |
| 6 | prereposetup | |
| 7 | postreposetup | sack |
| 8 | exclude | resolved |
| 9 | preresolve | |
| 10 | postresolve | resolved but no re-resolve |
| 11 | pretrans | pre_transaction |
| 12 | postrans | transaction |
| 13 | close | transaction |
| 14 | clean | |

Feel free to file a RFE for missing functionality if you need it.

CHAPTER 12

# Changes in DNF-2 compared to DNF-1

## CLI changes

### Reintroduction of YUM's configuration options `includepkgs` and `excludepkgs`

Due to a better compatibility with YUM, configuration options `include` and `exclude` were replaced by the original options *includepkgs* and *excludepkgs*.

### DNF group install `--with-optional` option

Installation of optional packages of group is changed from subcommand `with-optional` to option `--with-optional`.

## Python API changes

### All non-API methods and attributes are private

**Warning:** All non-API methods and attributes of *documented modules* are now private in order to accomplish more distinguishable API.

### Following API methods accept different arguments

1. *dnf.Base.add_remote_rpms()*
2. *dnf.Base.group_install()*
3. *dnf.cli.Command.configure()*
4. *dnf.cli.Command.run()*

5. *dnf.Plugin.read_config()*

DNF Plugins and components

- DNF Plugins Core
- DNF Plugins Extras
- Hawkey

Indices and tables

- genindex
- modindex
- search

# Python Module Index

## d

# Index

## Symbols



## A



## B



## C



## D

## E



## F



## G



## I



## K



## L



## M



## N

## V



## W