yum(8) - Linux man page

Name

yum - Yellowdog Updater Modified

Synopsis

yum [options] [command] [package ...]

Description

yum is an interactive, rpm based, package manager. It can automatically perform system updates, including dependency analysis and obsolete processing based on "repository" metadata. It can also perform installation of new packages, removal of old packages and perform queries on the installed and/or available packages among many other commands/services (see below). **yum** is similar to other high level package managers like apt-get and smart.

While there are some graphical interfaces directly to the **yum** code, more recent graphical interface development is happening with PackageKit and the gnome-packagekit application.

command is one of:
* install package1 [package2] [...]
* update [package1] [package2] [...]
* update-to [package1] [package2] [...]
* check-update
* upgrade [package1] [package2] [...]
* upgrade-to [package1] [package2] [...]
* distribution-synchronization [package1] [package2] [...]
* remove | erase package1 [package2] [...]
* list [...]
* info [...]
* provides | whatprovides feature1 [feature2] [...]
* clean [packages | metadata | expire-cache | rpmdb | plugins | all]
* makecache
* groupinstall group1 [group2] [...]
* groupupdate group1 [group2] [...]
* grouplist [hidden] [groupwildcard] [...]
* groupremove group1 [group2] [...]
* groupinfo group1 [...]
* search string1 [string2] [...]
* shell [filename]
* resolvedep dep1 [dep2] [...]
* localinstall rpmfile1 [rpmfile2] [...] (maintained for legacy reasons only - use install)

* localupdate rpmfile1 [rpmfile2] [...] (maintained for legacy reasons only - use update)

* reinstall package1 [package2] [...]
* downgrade package1 [package2] [...]
* deplist package1 [package2] [...]

* repolist [all|enabled|disabled]
* version [all | installed | available | group-* | nogroups* | grouplist | groupinfo]
* history [info|list|packages-list|packages-info|summary|addon-info|redo|undo|rollback|new|sync|stats]
* load-transaction [txfile]
* check
* help [command]

Unless the --help or -h option is given, one of the above commands must be present.

Repository configuration is honored in all operations.

install
Is used to install the latest version of a package or group of packages while ensuring that all dependencies are satisfied. (See **Specifying package names** for more information) If no package matches the given package **name**(s), they are assumed to be a shell glob and any matches are then installed. If the name starts with an @ character the rest of the name is used as though passed to the groupinstall command. If the name starts with a - character, then a search is done within the transaction and any matches are removed. If the name is a file, then install works like localinstall. If the name doesn't match a package, then package "provides" are searched (e.g. "_sqlitecache.so()(64bit)") as are filelists (Eg. "/usr/bin/yum"). Also note that for filelists, wildcards will match multiple packages.

update
If run without any packages, update will update every currently installed package. If one or more packages or package globs are specified, Yum will only update the listed packages. While updating packages, **yum** will ensure that all dependencies are satisfied. (See **Specifying package names** for more information) If the packages or globs specified match to packages which are not currently installed then update will not install them. update operates on groups, files, provides and filelists just like the "install" command.

If the main obsoletes configure option is true (default) or the --obsoletes flag is present **yum** will include package obsoletes in its calculations - this makes it better for distro-version changes, for example: upgrading from somelinux 8.0 to somelinux 9.

Note that "**update**" works on installed packages first, and only if there are no matches does it look for available packages. The difference is most noticeable when you do "**update** foo-1-2" which will act exactly as "**update** foo" if foo-1-2 is installed. You can use the "**update-to**" if you'd prefer that nothing happen in the above case.

update-to
This command works like "**update**" but always specifies the version of the package we want to update to.

check-update
Implemented so you could know if your machine had any updates that needed to be applied without running it interactively. Returns exit value of 100 if there are packages available for an update. Also returns a list of the packages to be updated in list format. Returns 0 if no packages are available for update. Returns 1 if an error occurred. Running in verbose mode also shows obsoletes.

upgrade
Is the same as the update command with the --obsoletes flag set. See update for more details.

upgrade-to
This command works like "**upgrade**" but always specifies the version of the package we want to update to.

distribution-synchronization or **distro-sync**
Synchronizes the installed package set with the latest packages available, this is done by either obsoleting, upgrading or downgrading as appropriate. This will "normally" do the same thing as the upgrade command however if you have the package FOO installed at version 4, and the latest available is only version 3, then this command will **downgrade** FOO to version 3.

This command does not perform operations on groups, local packages or negative selections.

remove or **erase**
Are used to remove the specified packages from the system as well as removing any packages which depend on the package being removed. remove operates on groups, files, provides and filelists just like the "install" command.(See **Specifying package names** for more information)

Note that "yum" is included in the protected_packages configuration, by default. So you can't accidentally remove yum itself.

list
Is used to list various information about available packages; more complete details are available in the *List Options* section below.

provides or **whatprovides**
Is used to find out which package provides some feature or file. Just use a specific name or a file-glob-syntax wildcards to list the packages available or installed that provide that feature or file.

search
This is used to find packages when you know something about the package but aren't sure of it's name. By default search will try searching just package names and summaries, but if that "fails" it will then try descriptions and url.

Yum search orders the results so that those packages matching more terms will appear first.

You can force searching everything by specifying "all" as the first argument.

info
Is used to list a description and summary information about available packages; takes the same arguments as in the *List Options* section below.

clean
Is used to clean up various things which accumulate in the **yum** cache directory over time. More complete details can be found in the *Clean Options* section below.

makecache
Is used to download and make usable all the metadata for the currently enabled **yum** repos.

groupinstall
Is used to install all of the individual packages in a group, of the specified types (this works as if you'd taken each of those package names and put them on the command line for a "yum install" command). The group_package_types configuration option specifies which types will be installed.

groupupdate

Is just an alias for groupinstall, which will do the right thing because "yum install X" and "yum update X" do the same thing, when X is already installed.

grouplist

Is used to list the available groups from all **yum** repos. Groups are marked as "installed" if all mandatory packages are installed, or if a group doesn't have any mandatory packages then it is installed if any of the optional or default package are installed. The optional "hidden" argument will also list groups marked as not being "user visible". If you pass the -v option, to enable verbose mode, then the groupids are displayed.

groupremove

Is used to remove all of the packages in a group, unlike "groupinstall" this will remove everything regardless of group_package_types. It is worth pointing out that packages can be in more than one group, so "groupinstall X Y" followed by "groupremove Y" does not do give you the same result as "groupinstall X".

The groupremove_leaf_only configuration changes the behaviour of this command to only remove packages which aren't required by something else.

groupinfo

Is used to give the description and package list of a group (and which type those packages are marked as). Note that you can use the yum-filter-data and yum-list-data plugins to get/use the data the other way around (Ie. what groups own packages need updating). If you pass the -v option, to enable verbose mode, then the package names are matched against installed/available packages similar to the list command.

shell

Is used to enter the 'yum shell', when a filename is specified the contents of that file is executed in yum shell mode. See ***yum-shell****(8)* for more info

resolvedep

Is used to list packages providing the specified dependencies, at most one package is listed per dependency.

localinstall

Is used to install a set of local rpm files. If required the enabled repositories will be used to resolve dependencies. Note that the install command will do a local install, if given a filename. This option is maintained for legacy reasons only.

localupdate

Is used to update the system by specifying local rpm files. Only the specified rpm files of which an older version is already installed will be installed, the remaining specified packages will be ignored. If required the enabled repositories will be used to resolve dependencies. Note that the update command will do a local update, if given a filename. This option is maintained for legacy reasons only.

reinstall

Will reinstall the identically versioned package as is currently installed. This does not work for "installonly" packages, like Kernels. reinstall operates on groups, files, provides and filelists just like the "install" command.

downgrade

Will try and downgrade a package from the version currently installed to the previously highest version (or the specified version). The depsolver will not necessarily work, but if you specify all the packages it should work (and thus. all the simple cases will work). Also this does not work

for "installonly" packages, like Kernels. downgrade operates on groups, files, provides, filelists and rpm files just like the "install" command.

deplist

Produces a list of all dependencies and what packages provide those dependencies for the given packages.

repolist

Produces a list of configured repositories. The default is to list all enabled repositories. If you pass -v, for verbose mode, more information is listed. If the first argument is 'enabled', 'disabled' or 'all' then the command will list those types of repos.

You can pass repo id or name arguments, or wildcards which to match against both of those. However if the id or name matches exactly then the repo will be listed even if you are listing enabled repos. and it is disabled.

In non-verbose mode the first column will start with a '*' if the repo. has metalink data and the latest metadata is not local. For non-verbose mode the last column will also display the number of packages in the repo. and (if there are any user specified excludes) the number of packages excluded.

One last special feature of repolist, is that if you are in non-verbose mode then yum will ignore any repo errors and output the information it can get (Eg. "yum clean all; yum -C repolist" will output something, although the package counts/etc. will be zeroed out).

version

Produces a "version" of the rpmdb, and of the enabled repositories if "all" is given as the first argument. You can also specify version groups in the version-groups configuration file. If you pass -v, for verbose mode, more information is listed. The version is calculated by taking an SHA1 hash of the packages (in sorted order), and the checksum_type/checksum_data entries from the yumdb. Note that this rpmdb version is now also used significantly within yum (esp. in yum history).

The version command will now show "groups" of packages as a separate version, and so takes sub-commands:

"version grouplist" - List the defined version groups.

"version groupinfo" - Get the complete list of packages within one or more version groups.

"version installed" - This is the default, only show the version information for installed packages.

"version available" - Only show the version information for available packages.

"version all" - Show the version information for installed and available packages.

"version nogroups | nogroups-*" - Just show the main version information.

"version group-*" - Just show the grouped version information, if more arguments are given then only show the data for those groups.

history

The history command allows the user to view what has happened in past transactions (assuming the history_record config. option is set). You can use info/list/packages-list/packages-info/summary to view what happened, undo/redo/rollback to act on that information and new to start a new history file.

The info/list/summary commands take either a transaction id or a package (with wildcards, as in **Specifying package names**), all three can also be passed no arguments. list can be passed the keyword "all" to list all the transactions.

The packages-list/packages-info commands takes a package (with wildcards, as in **Specifying package names**). And show data from the point of view of that package.

The undo/redo/rollback commands take either a single transaction id or the keyword last and an offset from the last transaction (Eg. if you've done 250 transactions, "last" refers to transaction 250, and "last-4" refers to transaction 246).

The undo/redo commands act on the specified transaction, undo'ing or repeating the work of that transaction. While the rollback command will undo all transactions up to the point of the specified transaction. For example, if you have 3 transactions, where package A; B and C where installed respectively. Then "undo 1" will try to remove package A, "redo 1" will try to install package A (if it is not still installed), and "rollback 1" will try to remove packages B and C. Note that after a "rollback 1" you will have a fourth transaction, although the ending rpmdb version (see: yum version) should be the same in transactions 1 and 4.

The addon-info command takes a transaction ID, and the packages-list command takes a package (with wildcards).

The stats command shows some statistics about the current history DB.

The sync commands allows you to change the rpmdb/yumdb data stored for any installed packages, to whatever is in the current rpmdb/yumdb (this is mostly useful when this data was not stored when the package went into the history DB).

In "history list" you can change the behaviour of the 2nd column via. the configuration option history_list_view.

In "history list" output the Altered column also gives some extra information if there was something not good with the transaction (this is also shown at the end of the package column in the packages-list command).

> - The rpmdb was changed, outside yum, after the transaction.
< - The rpmdb was changed, outside yum, before the transaction.
***** - The transaction aborted before completion.
- The transaction completed, but with a non-zero status.
E - The transaction completed fine, but had warning/error output during the transaction.
P - The transaction completed fine, but problems already existed in the rpmdb.
s - The transaction completed fine, but --skip-broken was enabled and had to skip some packages.

load-transaction

This command will re-load a saved yum transaction file, this allows you to run a transaction on one machine and then use it on another. The two common ways to get a saved yum transaction file are from "yum -q history addon-info last saved_tx" or via. the automatic saves in $TMPDIR/yum_save_tx.* when a transaction is solved but not run.

check

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

The info command can also take ranges of transaction ids, of the form start..end, which will then display a merged history as if all the transactions in the range had happened at once. Eg. "history info 1..4" will merge the first four transactions and display them as a single transaction.

help

Produces help, either for all commands or if given a command name then the help for that particular command.

General Options

Most command line options can be set using the configuration file as well and the descriptions indicate the necessary configuration option to set.

-h, --help

Help; display a help message and then quit.

-y, --assumeyes

Assume yes; assume that the answer to any question which would be asked is yes.
Configuration Option: **assumeyes**

-c, --config=[config file]

Specifies the config file location - can take HTTP and FTP URLs and local file paths.

-q, --quiet

Run without output. Note that you likely also want to use -y.

-v, --verbose

Run with a lot of debugging output.

-d, --debuglevel=[number]

Sets the debugging level to [number] - turns up or down the amount of things that are printed.
Practical range: 0 - 10
Configuration Option: **debuglevel**

-e, --errorlevel=[number]

Sets the error level to [number] Practical range 0 - 10. 0 means print only critical errors about which you must be told. 1 means print all errors, even ones that are not overly important. 1+ means print more errors (if any) -e 0 is good for cron jobs.
Configuration Option: **errorlevel**

--rpmverbosity=[name]

Sets the debug level to [name] for rpm scriptlets. 'info' is the default, other options are: 'critical', 'emergency', 'error', 'warn' and 'debug'.
Configuration Option: **rpmverbosity**

-R, --randomwait=[time in minutes]

Sets the maximum amount of time yum will wait before performing a command - it randomizes over the time.

-C, --cacheonly
Tells yum to run entirely from system cache - does not download or update any headers unless it has to to perform the requested action. If you're using this as a user yum will not use the tempcache for the user but will only use the system cache in the system cachedir.

--version
Reports the **yum** version number and installed package versions for everything in history_record_packages (can be added to by plugins).

--showduplicates
Doesn't limit packages to their latest versions in the info, list and search commands (will also affect plugins which use the doPackageLists() API).

--installroot=root
Specifies an alternative installroot, relative to which all packages will be installed.
Configuration Option: **installroot**

--enablerepo=repoidglob
Enables specific repositories by id or glob that have been disabled in the configuration file using the enabled=0 option.
Configuration Option: **enabled**

--disablerepo=repoidglob
Disables specific repositories by id or glob.
Configuration Option: **enabled**

--obsoletes
This option only has affect for an update, it enables **yum**'s obsoletes processing logic. For more information see the **update** command above.
Configuration Option: **obsoletes**

-x, --exclude=package
Exclude a specific package by name or glob from updates on all repositories. Configuration Option: **exclude**

--color=[always|auto|never]
Display colorized output automatically, depending on the output terminal, always (using ANSI codes) or never. Note that some commands (Eg. list and info) will do a little extra work when color is enabled. Configuration Option: **color**

--disableexcludes=[all|main|repoid]
Disable the excludes defined in your config files. Takes one of three options:
all == disable all excludes
main == disable excludes defined in [main] in yum.conf
repoid == disable excludes defined for that repo

--disableplugin=plugin
Run with one or more plugins disabled, the argument is a comma separated list of wildcards to match against plugin names.

--noplugins
Run with all plugins disabled.
Configuration Option: **plugins**

--nogpgcheck
Run with GPG signature checking disabled.
Configuration Option: **gpgcheck**

--skip-broken
Resolve depsolve problems by removing packages that are causing problems from the transaction.

Configuration Option: **skip_broken**

--releasever=version
Pretend the current release version is the given string. This is very useful when combined with --installroot. Note that with the default upstream cachedir, of /var/cache/yum, using this option will corrupt your cache (and you can use $releasever in your cachedir configuration to stop this).

-t, --tolerant
This option currently does nothing.

--setopt=option=value
Set any config option in yum config or repo files. For options in the global config just use: --setopt=option=value for repo options use: --setopt=repoid.option=value

List Options

The following are the ways which you can invoke **yum** in list mode. Note that all **list** commands include information on the version of the package.

OUTPUT

The format of the output of yum list is:

name.arch [epoch:]version-release repo or @installed-from-repo

yum list [all | glob_exp1] [glob_exp2] [...]
List all available and installed packages.

yum list available [glob_exp1] [...]
List all packages in the yum repositories available to be installed.

yum list updates [glob_exp1] [...]
List all packages with updates available in the yum repositories.

yum list installed [glob_exp1] [...]
List the packages specified by *args*. If an argument does not match the name of an available package, it is assumed to be a shell-style glob and any matches are printed.

yum list extras [glob_exp1] [...]
List the packages installed on the system that are not available in any yum repository listed in the config file.

yum list obsoletes [glob_exp1] [...]
List the packages installed on the system that are obsoleted by packages in any yum repository listed in the config file.

yum list recent
List packages recently added into the repositories. This is often not helpful, but what you may really want to use is "yum list-updateinfo new" from the security yum plugin.

Specifying Package Names

A package can be referred to for install, update, remove, list, info etc with any of the following as well as globs of any of the following:

name

name.arch

name-ver
name-ver-rel
name-ver-rel.arch
name-epoch:ver-rel.arch
epoch:name-ver-rel.arch

For example: **yum remove kernel-2.4.1-10.i686**
this will remove this specific kernel-ver-rel.arch.

Or: **yum list available 'foo*'**
will list all available packages that match 'foo*'. (The single quotes will keep your shell from expanding the globs.)

Clean Options

The following are the ways which you can invoke **yum** in clean mode. Note that "all files" in the commands below means "all files in currently enabled repositories". If you want to also clean any (temporarily) disabled repositories you need to use **--enablerepo='*'** option.

yum clean expire-cache
Eliminate the local data saying when the metadata and mirrorlists were downloaded for each repo. This means yum will revalidate the cache for each repo. next time it is used. However if the cache is still valid, nothing significant was deleted.

yum clean packages
Eliminate any cached packages from the system. Note that packages are not automatically deleted after they are downloaded.

yum clean headers
Eliminate all of the header files, which old versions of yum used for dependency resolution.

yum clean metadata
Eliminate all of the files which yum uses to determine the remote availability of packages. Using this option will force yum to download all the metadata the next time it is run.

yum clean dbcache
Eliminate the sqlite cache used for faster access to metadata. Using this option will force yum to download the sqlite metadata the next time it is run, or recreate the sqlite metadata if using an older repo.

yum clean rpmdb
Eliminate any cached data from the local rpmdb.

yum clean plugins
Tell any enabled plugins to eliminate their cached data.

yum clean all
Does all of the above.

Plugins

Yum can be extended through the use of plugins. A plugin is a Python ".py" file which is installed in one of the directories specified by the **pluginpath** option in yum.conf. For a plugin to work, the following conditions must be met:

1. The plugin module file must be installed in the plugin path as just described.

2. The global **plugins** option in /etc/yum.conf must be set to '1'.

3. A configuration file for the plugin must exist in /etc/yum/pluginconf.d/<plugin_name>.conf and the **enabled** setting in this file must set to '1'. The minimal content for such a configuration file is:

```
[main]
    enabled = 1
```

See the ***yum.conf(5)*** man page for more information on plugin related configuration options.

Files

```
/etc/yum.conf
/etc/yum/version-groups.conf
/etc/yum.repos.d/
/etc/yum/pluginconf.d/
/var/cache/yum/
```

See Also

pkcon (1)yum.conf (5)yum-updatesd (8)package-cleanup (1)repoquery (1)yum-complete-transaction (1)yumdownloader (1)yum-utils (1)yum-security (8)http://yum.baseurl.org/
http://yum.baseurl.org/wiki/Faq
yum search yum

Authors

See the Authors file included with this program.

Bugs

There of course aren't any bugs, but if you find any, you should first consult the FAQ mentioned above and then email the mailing list: yum@lists.baseurl.org or filed in bugzilla.

Referenced By

febootstrap(8), **mock**(1), **pirut**(8), **pup**(8), **puplet**(8), **system-cdinstall-helper**(8), **system-install-packages**(8), **yum-updatesd**(8), **yumdownloader**(1)