

Zypper package manager

Publication Date: 27 Sep 2024

Contents

- 1 Environment 2
- 2 What is Zypper? 2
- 3 How does Zypper work? 2
- 4 Benefits of Zypper 2
- 5 General usage 3
- 6 Using Zypper subcommands 4
- 7 Related topics 5

1 Environment

This document applies to the following products and product versions:

- SUSE Linux Enterprise Server 15 SP5, 15 SP4, 15 SP3, 15 SP2, 12 SP5
- SUSE Linux Enterprise Server for SAP Applications 15 SP5, 15 SP4, 15 SP3, 15 SP2, 12 SP5
- SUSE Linux Enterprise High Availability 15 SP5, 15 SP4, 15 SP3, 15 SP2, 12 SP5
- SUSE Linux Enterprise High Performance Computing 15 SP5, 15 SP4, 15 SP3, 15 SP2
- SUSE Linux Enterprise Desktop 15 SP5
- SUSE Linux Enterprise Real Time 15 SP5

2 What is Zypper?

Zypper is a command-line package manager for installing, updating and removing packages. It can also be used to manage repositories.

3 How does Zypper work?

Zypper works and behaves as a regular command-line tool. It features subcommands, arguments, and options that can be used to perform specific tasks.

4 Benefits of Zypper

Zypper offers several benefits compared to graphical package managers.

- Being a command-line tool, Zypper is faster in use and light on resources.
- Zypper actions can be scripted.
- Zypper can be used on systems that do not have graphical desktop environments. This makes it suitable for use with servers and remote machines.

5 General usage

The general syntax of Zypper is:

```
zypper [GLOBAL_OPTIONS] SUBCOMMAND [SUBCOMMAND_OPTIONS] [ARGUMENTS]
```

The components enclosed in brackets are not required. See **zypper help** for a list of general options and all subcommands. To get help for a specific subcommand, type **zypper help SUBCOMMAND**.

Zypper subcommands

The simplest way to execute Zypper is to type its name, followed by a command. For example, to apply all needed patches to the system, use:

```
> sudo zypper patch
```

Global options

Additionally, you can choose from one or more global options by typing them immediately before the command:

```
> sudo zypper --non-interactive patch
```

In the above example, the option --non-interactive means that the command is run without asking anything (automatically applying the default answers).

Command-specific options

To use options that are specific to a particular command, type them immediately after the command:

```
> sudo zypper patch --auto-agree-with-licenses
```

In the above example, --auto-agree-with-licenses is used to apply all needed patches to a system without you being asked to confirm any licenses. Instead, license will be accepted automatically.

Arguments

Some commands require one or more arguments. For example, when using the command **install**, you need to specify one or more packages you want to *install*:

```
> sudo zypper install mplayer
```

Some options also require a single argument. The following command will list all known patterns:

```
> zypper search --type pattern
```

You can combine all of the above. For example, the following command will install the `mc` and `vim` packages from the `factory` repository while being verbose:

```
> sudo zypper --verbose install --from factory mc vim
```

The `--from` option keeps all repositories enabled (for solving any dependencies) while requesting the package from the specified repository. The option `--repo` is an alias for `--from`, and you may use either one.

Most Zypper commands have a `--dry-run` option that does a simulation of the given command. It can be used for test purposes:

```
> sudo zypper remove --dry-run MozillaFirefox
```

Zypper supports the global `--userdata STRING` option. You can specify a string with this option, which gets written to Zypper's log files and plug-ins (such as the Btrfs plug-in). It can be used to mark and identify transactions in log files.

```
> sudo zypper --userdata STRING patch
```

6 Using Zypper subcommands

Zypper subcommands are executables that are stored in the directory `/usr/lib/zypper/commands`. If a subcommand is not found in this directory, Zypper automatically searches the rest of your `$PATH` for it. This enables writing your own local extensions and storing them in userspace. Executing subcommands in the Zypper shell, and using global Zypper options are not supported. List your available subcommands:

```
> zypper help subcommand
[...]
Available zypper subcommands in '/usr/lib/zypper/commands'

  appstream-cache
  lifecycle
  migration
  search-packages

Zypper subcommands available from elsewhere on your $PATH

<none>
```

View the help screen for a subcommand:

```
> zypper help appstream-cache
```

7 Related topics

- Installing and removing software with Zypper
- Updating software with Zypper