
## *Rocky Enterprise Linux 9.2 Manual Pages on command 'sources.list.5'*

***C:\>man sources.list.5***

SOURCES.LIST(5) APT SOURCES.LIST(5)

NAME

sources.list - List of configured APT data sources

DESCRIPTION

The source list /etc/apt/sources.list and the files contained in /etc/apt/sources.list.d/ are designed to support any number of active sources and a variety of source media. The files list one source per line (one-line style) or contain multiline stanzas defining one or more sources per stanza (deb822 style), with the most preferred source listed first (in case a single version is available from more than one source). The information available from the configured sources is acquired by apt-get update (or by an equivalent command from another APT front-end).

SOURCES.LIST.D

The /etc/apt/sources.list.d directory provides a way to add sources.list entries in separate files. Two different file formats are allowed as described in the next two sections. Filenames need to have either the extension .list or .sources depending on the contained format. The filenames may only contain letters (a-z and A-Z), digits (0-9), underscore (_), hyphen (-) and period (.) characters. Otherwise APT will print a notice that it has ignored a file, unless that file matches a pattern in the Dir::Ignore-Files-Silently configuration list - in which case it will be silently ignored.

ONE-LINE-STYLE FORMAT

Files in this format have the extension .list. Each line specifying a source starts with a type (e.g. deb-src) followed by options and arguments for this type. Individual entries cannot be continued onto a following line. Empty lines are ignored, and a # character anywhere on a line marks the remainder of that line as a comment. Consequently an entry can be disabled by commenting out the entire line. If options should be provided they are separated by spaces and all of them together are enclosed by square brackets ([]) included in the line after the type separated from it with a space. If an option allows multiple values these are separated from each other with a comma (,). An option name is separated from its value(s) by an equals sign (=). Multivalue options also have -= and += as separators, which instead of replacing the default with the given value(s) modify the default value(s) to remove or include the given values.

This is the traditional format and supported by all apt versions. Note that not all options as described below are supported by all apt versions. Note also that some older applications parsing this format on their own might not expect to encounter options as they were uncommon before the introduction of multi-architecture support.

DEB822-STYLE FORMAT

Files in this format have the extension .sources. The format is similar in syntax to other files used by Debian and its derivatives, such as the metadata files that apt will download from the configured sources or the debian/control file in a Debian source package. Individual entries are separated by an empty line; additional empty lines are ignored, and a # character at the start of the line marks the entire line as a comment. An entry can hence be disabled by commenting out each line belonging to the stanza, but it is usually easier to add the field "Enabled: no" to the stanza to disable the entry. Removing the field or setting it to yes re-enables it. Options have the same syntax as every other field: A field name separated by a colon (:) and optionally spaces from its value(s). Note especially that multiple values are separated by whitespaces (like spaces, tabs and newlines), not by commas as in the one-line format. Multivalue fields like Architectures also have Architectures-Add and Architectures-Remove to modify the default value rather than replacing it.

This is a new format supported by apt itself since version 1.1. Previous versions

ignore such files with a notice message as described earlier. It is intended to make this format gradually the default format, deprecating the previously described one-line-style format, as it is easier to create, extend and modify for humans and machines alike especially if a lot of sources and/or options are involved.

Developers who are working with and/or parsing apt sources are highly encouraged to add support for this format and to contact the APT team to coordinate and share this work. Users can freely adopt this format already, but may encounter problems with software not supporting the format yet.

## THE DEB AND DEB-SRC TYPES: GENERAL FORMAT

The deb type references a typical two-level Debian archive, distribution/component. The distribution is generally a suite name like stable or testing or a codename like buster or bullseye while component is one of main, contrib or non-free. The deb-src type references a Debian distribution's source code in the same form as the deb type. A deb-src line is required to fetch source indexes.

The format for two one-line-style entries using the deb and deb-src types is:

```
deb [ option1=value1 option2=value2 ] uri suite [component1] [component2] [...]
deb-src [ option1=value1 option2=value2 ] uri suite [component1] [component2] [...]
```

Alternatively the equivalent entry in deb822 style looks like this:

```
Types: deb deb-src
URIs: uri
Suites: suite
Components: [component1] [component2] [...]
option1: value1
option2: value2
```

The URI for the deb type must specify the base of the Debian distribution, from which APT will find the information it needs.  suite can specify an exact path, in which case the components must be omitted and suite must end with a slash (/). This is useful for the case when only a particular sub-directory of the archive denoted by the URI is of interest. If suite does not specify an exact path, at least one component must be present.

suite may also contain a variable, $(ARCH) which expands to the Debian architecture (such as amd64 or armel) used on the system. This permits architecture-independent sources.list files to be used. In general this is only of interest when specifying

an exact path; APT will automatically generate a URI with the current architecture otherwise.

Especially in the one-line-style format since only one distribution can be specified per line it may be necessary to have multiple lines for the same URI, if a subset of all available distributions or components at that location is desired. APT will sort the URI list after it has generated a complete set internally, and will collapse multiple references to the same Internet host, for instance, into a single connection, so that it does not inefficiently establish a connection, close it, do something else, and then re-establish a connection to that same host. APT also parallelizes connections to different hosts to more effectively deal with sites with low bandwidth.

It is important to list sources in order of preference, with the most preferred source listed first. Typically this will result in sorting by speed from fastest to slowest (CD-ROM followed by hosts on a local network, followed by distant Internet hosts, for example).

As an example, the sources for your distribution could look like this in one-line-style format:

```
deb http://us.archive.ubuntu.com/ubuntu focal main restricted
deb http://security.ubuntu.com/ubuntu focal-security main restricted
deb http://us.archive.ubuntu.com/ubuntu focal-updates main restricted
```

or like this in deb822 style format:

```
Types: deb
URIs: http://us.archive.ubuntu.com/ubuntu
Suites: focal focal-updates
Components: main restricted
Types: deb
URIs: http://security.ubuntu.com/ubuntu
Suites: focal-security
Components: main restricted
```

THE DEB AND DEB-SRC TYPES: OPTIONS

Each source entry can have options specified to modify which source is accessed and how data is acquired from it. Format, syntax and names of the options vary between the one-line-style and deb822-style formats as described, but they both have the

same options available. For simplicity we list the deb822 field name and provide the one-line name in brackets. Remember that besides setting multivalue options explicitly, there is also the option to modify them based on the default, but we aren't listing those names explicitly here. Unsupported options are silently ignored by all APT versions.

? Architectures (arch) is a multivalue option defining for which architectures information should be downloaded. If this option isn't set the default is all architectures as defined by the APT::Architectures config option.

? Languages (lang) is a multivalue option defining for which languages information such as translated package descriptions should be downloaded. If this option isn't set the default is all languages as defined by the Acquire::Languages config option.

? Targets (target) is a multivalue option defining which download targets apt will try to acquire from this source. If not specified, the default set is defined by the Acquire::IndexTargets configuration scope (targets are specified by their name in the Created-By field). Additionally, targets can be enabled or disabled by using the Identifier field as an option with a boolean value instead of using this multivalue option.

? PDiffs (pdiffs) is a yes/no value which controls if APT should try to use PDiffs to update old indexes instead of downloading the new indexes entirely. The value of this option is ignored if the repository doesn't announce the availability of PDiffs. Defaults to the value of the option with the same name for a specific index file defined in the Acquire::IndexTargets scope, which itself defaults to the value of configuration option Acquire::PDiffs which defaults to yes.

? By-Hash (by-hash) can have the value yes, no or force and controls if APT should try to acquire indexes via a URI constructed from a hashsum of the expected file instead of using the well-known stable filename of the index. Using this can avoid hashsum mismatches, but requires a supporting mirror. A yes or no value activates/disables the use of this feature if this source indicates support for it, while force will enable the feature regardless of what the source indicates. Defaults to the value of the option of the same name for a specific index file defined in the Acquire::IndexTargets scope, which

itself defaults to the value of configuration option Acquire::By-Hash which defaults to yes.

Furthermore, there are options which if set affect all sources with the same URI and Suite, so they have to be set on all such entries and can not be varied between different components. APT will try to detect and error out on such anomalies.

? Allow-Insecure (allow-insecure), Allow-Weak (allow-weak) and Allow-Downgrade-To-Insecure (allow-downgrade-to-insecure) are boolean values which all default to no. If set to yes they circumvent parts of apt-secure(8) and should therefore not be used lightly!

? Trusted (trusted) is a tri-state value which defaults to APT deciding if a source is considered trusted or if warnings should be raised before e.g. packages are installed from this source. This option can be used to override that decision. The value yes tells APT always to consider this source as trusted, even if it doesn't pass authentication checks. It disables parts of apt-secure(8), and should therefore only be used in a local and trusted context (if at all) as otherwise security is breached. The value no does the opposite, causing the source to be handled as untrusted even if the authentication checks passed successfully. The default value can't be set explicitly.

? Signed-By (signed-by) is an option to require a repository to pass apt-secure(8) verification with a certain set of keys rather than all trusted keys apt has configured. It is specified as a list of absolute paths to keyring files (have to be accessible and readable for the _apt system user, so ensure everyone has read-permissions on the file) and fingerprints of keys to select from these keyrings. If no keyring files are specified the default is the trusted.gpg keyring and all keyrings in the trusted.gpg.d/ directory (see apt-key fingerprint). If no fingerprint is specified all keys in the keyrings are selected. A fingerprint will accept also all signatures by a subkey of this key, if this isn't desired an exclamation mark (!) can be appended to the fingerprint to disable this behaviour. The option defaults to the value of the option with the same name if set in the previously acquired Release file of this repository (only fingerprints can be specified there through). Otherwise all keys in the trusted keyrings are considered valid signers for this repository.

? Check-Valid-Until (check-valid-until) is a yes/no value which controls if APT should try to detect replay attacks. A repository creator can declare a time until which the data provided in the repository should be considered valid, and if this time is reached, but no new data is provided, the data is considered expired and an error is raised. Besides increasing security, as a malicious attacker can't send old data forever to prevent a user from upgrading to a new version, this also helps users identify mirrors which are no longer updated. However, some repositories such as historic archives are not updated any more by design, so this check can be disabled by setting this option to no. Defaults to the value of configuration option Acquire::Check-Valid-Until which itself defaults to yes.

? Valid-Until-Min (valid-until-min) and Valid-Until-Max (valid-until-max) can be used to raise or lower the time period in seconds in which the data from this repository is considered valid. -Max can be especially useful if the repository provides no Valid-Until field on its Release file to set your own value, while -Min can be used to increase the valid time on seldom updated (local) mirrors of a more frequently updated but less accessible archive (which is in the sources.list as well) instead of disabling the check entirely. Default to the value of the configuration options Acquire::Min-ValidTime and Acquire::Max-ValidTime which are both unset by default.

? Check-Date (check-date) is a yes/no value which controls if APT should consider the machine's time correct and hence perform time related checks, such as verifying that a Release file is not from the future. Disabling it also disables the Check-Valid-Until option mentioned above.

? Date-Max-Future (date-max-future) controls how far from the future a repository may be. Default to the value of the configuration option Acquire::Max-FutureTime which is 10 seconds by default.

? InRelease-Path (inrelease-path) determines the path to the InRelease file, relative to the normal position of an InRelease file. By default, this option is unset and APT will try to fetch an InRelease or, if that fails, a Release file and its associated Release.gpg file. By setting this option, the specified path will be tried instead of the InRelease file, and the fallback to Release files will be disabled.

## URI SPECIFICATION

The currently recognized URI types are:

### http (apt-transport-http(1))

The http scheme specifies an HTTP server for an archive and is the most commonly used method. The URI can directly include login information if the archive requires it, but the use of apt_auth.conf(5) should be preferred. The method also supports SOCKS5 and HTTP(S) proxies either configured via apt-specific configuration or specified by the environment variable http_proxy in the format (assuming an HTTP proxy requiring authentication) http://user:pass@server:port/. The authentication details for proxies can also be supplied via apt_auth.conf(5).

Note that these forms of authentication are insecure as the whole communication with the remote server (or proxy) is not encrypted so a sufficiently capable attacker can observe and record login as well as all other interactions. The attacker can not modify the communication through as APT's data security model is independent of the chosen transport method. See apt-secure(8) for details.

### https (apt-transport-https(1))

The https scheme specifies an HTTPS server for an archive and is very similar in use and available options to the http scheme. The main difference is that the communication between apt and server (or proxy) is encrypted. Note that the encryption does not prevent an attacker from knowing which server (or proxy) apt is communicating with and deeper analysis can potentially still reveal which data was downloaded. If this is a concern the Tor-based schemes mentioned further below might be a suitable alternative.

### mirror, mirror+scheme (apt-transport-mirror(1))

The mirror scheme specifies the location of a mirrorlist. By default the scheme used for the location is http, but any other scheme can be used via mirror+scheme. The mirrorlist itself can contain many different URIs for mirrors the APT client can transparently pick, choose and fallback between intended to help both with distributing the load over the available mirrors and ensuring that clients can acquire data even if some configured mirrors are not available.

### file

The file scheme allows an arbitrary directory in the file system to be considered an archive. This is useful for NFS mounts and local mirrors or archives.

cdrom

The cdrom scheme allows APT to use a local CD-ROM, DVD or USB drive with media swapping. Use the apt-cdrom(8) program to create cdrom entries in the source list.

ftp

The ftp scheme specifies an FTP server for an archive. Use of FTP is on the decline in favour of http and https and many archives either never offered or are retiring FTP access. If you still need this method many configuration options for it are available in the Acquire::ftp scope and detailed in apt.conf(5).

Please note that an FTP proxy can be specified by using the ftp_proxy environment variable. It is possible to specify an HTTP proxy (HTTP proxy servers often understand FTP URLs) using this environment variable and only this environment variable. Proxies using HTTP specified in the configuration file will be ignored.

copy

The copy scheme is identical to the file scheme except that packages are copied into the cache directory instead of used directly at their location. This is useful for people using removable media to copy files around with APT.

rsh, ssh

The rsh/ssh method invokes RSH/SSH to connect to a remote host and access the files as a given user. Prior configuration of rhosts or RSA keys is recommended. The standard find and dd commands are used to perform the file transfers from the remote host.

adding more recognizable URI types

APT can be extended with more methods shipped in other optional packages, which should follow the naming scheme apt-transport-method. For instance, the APT team also maintains the package apt-transport-tor, which provides access methods for HTTP and HTTPS URIs routed via the Tor network.

EXAMPLES

Uses the archive stored locally (or NFS mounted) at /home/apt/debian for

stable/main, stable/contrib, and stable/non-free.

deb file:/home/apt/debian stable main contrib non-free

Types: deb

URIs: file:/home/apt/debian

Suites: stable

Components: main contrib non-free

As above, except this uses the unstable (development) distribution.

deb file:/home/apt/debian unstable main contrib non-free

Types: deb

URIs: file:/home/apt/debian

Suites: unstable

Components: main contrib non-free

Sources specification for the above.

deb-src file:/home/apt/debian unstable main contrib non-free

Types: deb-src

URIs: file:/home/apt/debian

Suites: unstable

Components: main contrib non-free

The first line gets package information for the architectures in APT::Architectures

while the second always retrieves amd64 and armel.

deb http://deb.debian.org/debian buster main

deb [ arch=amd64,armel ] http://deb.debian.org/debian buster main

Types: deb

URIs: http://deb.debian.org/debian

Suites: buster

Components: main

Types: deb

URIs: http://deb.debian.org/debian

Suites: buster

Components: main

Architectures: amd64 armel

Uses HTTP to access the archive at archive.debian.org, and uses only the hamm/main

area.

deb http://archive.debian.org/debian-archive hamm main

Types: deb

URIs: http://archive.debian.org/debian-archive

Suites: hamm

Components: main

Uses FTP to access the archive at ftp.debian.org, under the debian directory, and uses only the buster/contrib area.

deb ftp://ftp.debian.org/debian buster contrib

Types: deb

URIs: ftp://ftp.debian.org/debian

Suites: buster

Components: contrib

Uses FTP to access the archive at ftp.debian.org, under the debian directory, and uses only the unstable/contrib area. If this line appears as well as the one in the previous example in sources.list a single FTP session will be used for both resource lines.

deb ftp://ftp.debian.org/debian unstable contrib

Types: deb

URIs: ftp://ftp.debian.org/debian

Suites: unstable

Components: contrib

Uses HTTP to access the archive at ftp.tlh.debian.org, under the universe directory, and uses only files found under unstable/binary-i386 on i386 machines, unstable/binary-amd64 on amd64, and so forth for other supported architectures. [Note this example only illustrates how to use the substitution variable; official debian archives are not structured like this]

deb http://ftp.tlh.debian.org/universe unstable/binary-$(ARCH)/

Types: deb

URIs: http://ftp.tlh.debian.org/universe

Suites: unstable/binary-$(ARCH)/

Uses HTTP to get binary packages as well as sources from the stable, testing and unstable suites and the components main and contrib.

deb http://deb.debian.org/debian stable main contrib

deb-src http://deb.debian.org/debian stable main contrib

deb http://deb.debian.org/debian testing main contrib

deb-src http://deb.debian.org/debian testing main contrib

deb http://deb.debian.org/debian unstable main contrib

deb-src http://deb.debian.org/debian unstable main contrib

Types: deb deb-src

URIs: http://deb.debian.org/debian

Suites: stable testing unstable

Components: main contrib

## SEE ALSO

apt-get(8), apt.conf(5), /usr/share/doc/apt-doc/acquire-additional-files.md.gz

## BUGS

APT bug page[1]. If you wish to report a bug in APT, please see

/usr/share/doc/debian/bug-reporting.txt or the reportbug(1) command.

## AUTHORS

Jason Gunthorpe

APT team

## NOTES

1. APT bug page

   http://bugs.debian.org/src:apt