NAME

apt.conf − Configuration file for APT

DESCRIPTION

/etc/apt/apt.conf is the main configuration file shared by all the tools in the APT suite of tools, though it is by no means the only place options can be set. The suite also shares a common command line parser to provide a uniform environment.

When an APT tool starts up it will read the configuration files in the following order:

1. the file specified by the **APT_CONFIG** environment variable (if any)
2. all files in Dir::Etc::Parts in alphanumeric ascending order which have either no or "conf" as filename extension and which only contain alphanumeric, hyphen (−), underscore (_) 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.
3. the main configuration file specified by Dir::Etc::main
4. all options set in the binary specific configuration subtree are moved into the root of the tree.
5. the command line options are applied to override the configuration directives or to load even more configuration files.

SYNTAX

The configuration file is organized in a tree with options organized into functional groups. Option specification is given with a double colon notation; for instance APT::Get::Assume−Yes is an option within the APT tool group, for the Get tool. Options do not inherit from their parent groups.

Syntactically the configuration language is modeled after what the ISC tools such as bind and dhcp use. Lines starting with // are treated as comments (ignored), as well as all text between /* and */, just like C/C++ comments. Each line is of the form APT::Get::Assume−Yes "true";. The quotation marks and trailing semicolon are required. The value must be on one line, and there is no kind of string concatenation. Values must not include backslashes or extra quotation marks. Option names are made up of alphanumeric characters and the characters "/−:._+". A new scope can be opened with curly braces, like this:

```
APT {
  Get {
    Assume−Yes "true";
    Fix−Broken "true";
  };
};
```

with newlines placed to make it more readable. Lists can be created by opening a scope and including a single string enclosed in quotes followed by a semicolon. Multiple entries can be included, separated by a semicolon.

```
DPkg::Pre−Install−Pkgs {"/usr/sbin/dpkg−preconfigure −−apt";};
```

In general the sample configuration file /usr/share/doc/apt/examples/configure−index.gz is a good guide for how it should look.

Case is not significant in names of configuration items, so in the previous example you could use dpkg::pre−install−pkgs.

Names for the configuration items are optional if a list is defined as can be seen in the DPkg::Pre−Install−Pkgs example above. If you don't specify a name a new entry will simply add a new option to the list. If you specify a name you can override the option in the same way as any other option by reassigning a new value to the option.

Two special commands are defined: #include (which is deprecated and not supported by alternative implementations) and #clear. #include will include the given file, unless the filename ends in a slash, in which case the whole directory is included. #clear is used to erase a part of the configuration tree. The specified element and all its descendants are erased. (Note that these lines also need to end with a semicolon.)

The #clear command is the only way to delete a list or a complete scope. Reopening a scope (or using the syntax described below with an appended ::) will *not* override previously written entries. Options can only be overridden by addressing a new value to them – lists and scopes can't be overridden, only cleared.

All of the APT tools take an −o option which allows an arbitrary configuration directive to be specified on the command line. The syntax is a full option name (APT::Get::Assume−Yes for instance) followed by an equals sign then the new value of the option. To append a new element to a list, add a trailing :: to the name of the list. (As you might suspect, the scope syntax can't be used on the command line.)

Note that appending items to a list using :: only works for one item per line, and that you should not use it in combination with the scope syntax (which adds :: implicitly). Using both syntaxes together will trigger a bug which some users unfortunately depend on: an option with the unusual name "::" which acts like every other option with a name. This introduces many problems; for one thing, users who write multiple lines in this *wrong* syntax in the hope of appending to a list will achieve the opposite, as only the last assignment for this option "::" will be used. Future versions of APT will raise errors and stop working if they encounter this misuse, so please correct such statements now while APT doesn't explicitly complain about them.

THE APT GROUP

This group of options controls general APT behavior as well as holding the options for all of the tools.

Architecture

System Architecture; sets the architecture to use when fetching files and parsing package lists. The internal default is the architecture apt was compiled for.

Architectures

All Architectures the system supports. For instance, CPUs implementing the amd64 (also called x86−64) instruction set are also able to execute binaries compiled for the i386 (x86) instruction set. This list is used when fetching files and parsing package lists. The initial default is always the system's native architecture (APT::Architecture), and foreign architectures are added to the default list when they are registered via **dpkg −−add−architecture**.

Compressor

This scope defines which compression formats are supported, how compression and decompression can be performed if support for this format isn't built into apt directly and a cost−value indicating how costly it is to compress something in this format. As an example the following configuration stanza would allow apt to download and uncompress as well as create and store files with the low−cost .reversed file extension which it will pass to the command **rev** without additional commandline parameters for compression and uncompression:

```
APT::Compressor::rev {
        Name "rev";
        Extension ".reversed";
        Binary "rev";
        CompressArg {};
        UncompressArg {};
        Cost "10";
};
```

Build−Profiles

List of all build profiles enabled for build−dependency resolution, without the "profile." namespace prefix. By default this list is empty. The **DEB_BUILD_PROFILES** as used by **dpkg-buildpackage**(1) overrides the list notation.

Default−Release

Default release to install packages from if more than one version is available. Contains release name, codename or release version. Examples: 'stable', 'testing', 'unstable', 'buster', 'bullseye', '4.0', '5.0*'. See also **apt_preferences**(5).

Ignore−Hold
Ignore held packages; this global option causes the problem resolver to ignore held packages in its decision making.

Clean−Installed
Defaults to on. When turned on the autoclean feature will remove any packages which can no longer be downloaded from the cache. If turned off then packages that are locally installed are also excluded from cleaning − but note that APT provides no direct means to reinstall them.

Immediate−Configure
Defaults to on, which will cause APT to install essential and important packages as soon as possible in an install/upgrade operation, in order to limit the effect of a failing **dpkg**(1) call. If this option is disabled, APT treats an important package in the same way as an extra package: between the unpacking of the package A and its configuration there can be many other unpack or configuration calls for other unrelated packages B, C etc. If these cause the **dpkg**(1) call to fail (e.g. because package B's maintainer scripts generate an error), this results in a system state in which package A is unpacked but unconfigured − so any package depending on A is now no longer guaranteed to work, as its dependency on A is no longer satisfied.

The immediate configuration marker is also applied in the potentially problematic case of circular dependencies, since a dependency with the immediate flag is equivalent to a Pre−Dependency. In theory this allows APT to recognise a situation in which it is unable to perform immediate configuration, abort, and suggest to the user that the option should be temporarily deactivated in order to allow the operation to proceed. Note the use of the word "theory" here; in the real world this problem has rarely been encountered, in non−stable distribution versions, and was caused by wrong dependencies of the package in question or by a system in an already broken state; so you should not blindly disable this option, as the scenario mentioned above is not the only problem it can help to prevent in the first place.

Before a big operation like dist−upgrade is run with this option disabled you should try to explicitly install the package APT is unable to configure immediately; but please make sure you also report your problem to your distribution and to the APT team with the bug link below, so they can work on improving or correcting the upgrade process.

Force−LoopBreak
Never enable this option unless you *really* know what you are doing. It permits APT to temporarily remove an essential package to break a Conflicts/Conflicts or Conflicts/Pre−Depends loop between two essential packages. *Such a loop should never exist and is a grave bug*. This option will work if the essential packages are not **tar**, **gzip**, **libc**, **dpkg**, **dash** or anything that those packages depend on.

Cache−Start, **Cache−Grow**, **Cache−Limit**
APT uses since version 0.7.26 a resizable memory mapped cache file to store the available information. Cache−Start acts as a hint of the size the cache will grow to, and is therefore the amount of memory APT will request at startup. The default value is 20971520 bytes (~20 MB). Note that this amount of space needs to be available for APT; otherwise it will likely fail ungracefully, so for memory restricted devices this value should be lowered while on systems with a lot of configured sources it should be increased. Cache−Grow defines in bytes with the default of 1048576 (~1 MB) how much the cache size will be increased in the event the space defined by Cache−Start is not enough. This value will be applied again and again until either the cache is big enough to store all information or the size of the cache reaches the Cache−Limit. The default of Cache−Limit is 0 which stands for no limit. If Cache−Grow is set to 0 the automatic growth of the cache is disabled.

Build−Essential
Defines which packages are considered essential build dependencies.

Get
The Get subsection controls the **apt-get**(8) tool; please see its documentation for more information about the options here.

Cache
The Cache subsection controls the **apt-cache**(8) tool; please see its documentation for more information about the options here.

CDROM
The CDROM subsection controls the **apt-cdrom**(8) tool; please see its documentation for more information about the options here.

THE ACQUIRE GROUP

The Acquire group of options controls the download of packages as well as the various "acquire methods" responsible for the download itself (see also **sources.list**(5)).

Check−Date
Security related option defaulting to true, enabling time−related checks. Disabling it means that the machine's time cannot be trusted, and APT will hence disable all time−related checks, such as **Check−Valid−Until** and verifying that the Date field of a release file is not in the future.

Max−FutureTime
Maximum time (in seconds) before its creation (as indicated by the Date header) that the Release file should be considered valid. The default value is 10. Archive specific settings can be made by appending the label of the archive to the option name. Preferably, the same can be achieved for specific **sources.list**(5) entries by using the **Date−Max−Future** option there.

Check−Valid−Until
Security related option defaulting to true, as giving a Release file's validation an expiration date prevents replay attacks over a long timescale, and can also for example help users to identify mirrors that are no longer updated − but the feature depends on the correctness of the clock on the user system. Archive maintainers are encouraged to create Release files with the Valid−Until header, but if they don't or a stricter value is desired the Max−ValidTime option below can be used. The **Check−Valid−Until** option of **sources.list**(5) entries should be preferred to disable the check selectively instead of using this global override.

Max−ValidTime
Maximum time (in seconds) after its creation (as indicated by the Date header) that the Release file should be considered valid. If the Release file itself includes a Valid−Until header the earlier date of the two is used as the expiration date. The default value is 0 which stands for "valid forever". Archive specific settings can be made by appending the label of the archive to the option name. Preferably, the same can be achieved for specific **sources.list**(5) entries by using the **Valid−Until−Max** option there.

Min−ValidTime
Minimum time (in seconds) after its creation (as indicated by the Date header) that the Release file should be considered valid. Use this if you need to use a seldom updated (local) mirror of a more frequently updated archive with a Valid−Until header instead of completely disabling the expiration date checking. Archive specific settings can and should be used by appending the label of the archive to the option name. Preferably, the same can be achieved for specific **sources.list**(5) entries by using the **Valid−Until−Min** option there.

AllowTLS
Allow use of the internal TLS support in the http method. If set to false, this completely disables support for TLS in apt's own methods (excluding the curl−based https method). No TLS−related functions will be called anymore.

PDiffs
Try to download deltas called PDiffs for indexes (like Packages files) instead of downloading whole ones. True by default. Preferably, this can be set for specific **sources.list**(5) entries or index files by using the **PDiffs** option there.

Two sub−options to limit the use of PDiffs are also available: FileLimit can be used to specify a maximum number of PDiff files should be downloaded to update a file. SizeLimit on the other hand is the maximum percentage of the size of all patches compared to the size of the targeted file. If one of these limits is exceeded the complete file is downloaded instead of the patches.

By−Hash
Try to download indexes via an URI constructed from a hashsum of the expected file rather than downloaded via a well−known stable filename. True by default, but automatically disabled if the source indicates no support for it. Usage can be forced with the special value "force". Preferably, this can be set for specific **sources.list**(5) entries or index files by using the **By−Hash** option there.

Queue−Mode
Queuing mode; Queue−Mode can be one of host or access which determines how APT parallelizes outgoing connections. host means that one connection per target host will be opened, access means that one connection per URI type will be opened.

Retries
Number of retries to perform. If this is non−zero APT will retry failed files the given number of times.

Source−Symlinks
Use symlinks for source archives. If set to true then source archives will be symlinked when possible instead of copying. True is the default.

http https
The options in these scopes configure APT's acquire transports for the protocols HTTP and HTTPS and are documented in the **apt-transport-http**(1) and **apt-transport-https**(1) manpages respectively.

ftp
ftp::Proxy sets the default proxy to use for FTP URIs. It is in the standard form of ftp://[[user][:pass]@]host[:port]/. Per host proxies can also be specified by using the form ftp::Proxy::<host> with the special keyword DIRECT meaning to use no proxies. If no one of the above settings is specified, **ftp_proxy** environment variable will be used. To use an FTP proxy you will have to set the ftp::ProxyLogin script in the configuration file. This entry specifies the commands to send to tell the proxy server what to connect to. Please see /usr/share/doc/apt/examples/configure−index.gz for an example of how to do this. The substitution variables representing the corresponding URI component are $(PROXY_USER), $(PROXY_PASS), $(SITE_USER), $(SITE_PASS), $(SITE) and $(SITE_PORT).

The option timeout sets the timeout timer used by the method; this value applies to the connection as well as the data timeout.

Several settings are provided to control passive mode. Generally it is safe to leave passive mode on; it works in nearly every environment. However, some situations require that passive mode be disabled and port mode FTP used instead. This can be done globally or for connections that go through a proxy or for a specific host (see the sample config file for examples).

It is possible to proxy FTP over HTTP by setting the **ftp_proxy** environment variable to an HTTP URL − see the discussion of the http method above for syntax. You cannot set this in the configuration file and it is not recommended to use FTP over HTTP due to its low efficiency.

The setting ForceExtended controls the use of RFC2428 EPSV and EPRT commands. The default is false, which means these commands are only used if the control connection is IPv6. Setting this to true forces their use even on IPv4 connections. Note that most FTP servers do not support RFC2428.

cdrom
For URIs using the cdrom method, the only configurable option is the mount point, cdrom::Mount, which must be the mount point for the CD−ROM (or DVD, or whatever) drive as specified in /etc/fstab. It is possible to provide alternate mount and unmount commands if your mount point cannot be listed in the fstab. The syntax is to put

/cdrom/::Mount "foo";

within the cdrom block. It is important to have the trailing slash. Unmount commands can be specified using UMount.

gpgv
For GPGV URIs the only configurable option is gpgv::Options, which passes additional parameters to gpgv.

CompressionTypes
List of compression types which are understood by the acquire methods. Files like Packages can be available in various compression formats. By default the acquire methods can decompress and recompress many common formats like **xz** and **gzip**; with this scope the supported formats can be queried, modified as well as support for more formats added (see also **APT::Compressor**). The syntax for this is:

Acquire::CompressionTypes::*FileExtension* "*Methodname*";

Also, the Order subgroup can be used to define in which order the acquire system will try to download the compressed files. The acquire system will try the first and proceed with the next compression type in this list on error, so to prefer one over the other type simply add the preferred type first – types not already added will be implicitly appended to the end of the list, so e.g.

Acquire::CompressionTypes::Order:: "gz";

can be used to prefer **gzip** compressed files over all other compression formats. If **xz** should be preferred over **gzip** and **bzip2** the configure setting should look like this:

Acquire::CompressionTypes::Order { "xz"; "gz"; };

It is not needed to add bz2 to the list explicitly as it will be added automatically.

Note that the Dir::Bin::*Methodname* will be checked at run time. If this option has been set and support for this format isn't directly built into apt, the method will only be used if this file exists; e.g. for the bzip2 method (the inbuilt) setting is:

Dir::Bin::bzip2 "/bin/bzip2";

Note also that list entries specified on the command line will be added at the end of the list specified in the configuration files, but before the default entries. To prefer a type in this case over the ones specified in the configuration files you can set the option direct – not in list style. This will not override the defined list; it will only prefix the list with this type.

The special type uncompressed can be used to give uncompressed files a preference, but note that most archives don't provide uncompressed files so this is mostly only usable for local mirrors.

GzipIndexes
When downloading gzip compressed indexes (Packages, Sources, or Translations), keep them gzip compressed locally instead of unpacking them. This saves quite a lot of disk space at the expense of more CPU requirements when building the local package caches. False by default.

Languages
The Languages subsection controls which Translation files are downloaded and in which order APT tries to display the description–translations. APT will try to display the first available description in the language which is listed first. Languages can be defined with their short or long language codes. Note that not all archives provide Translation files for every language – the long language codes are especially rare.

The default list includes "environment" and "en". "environment" has a special meaning here: it will be replaced at runtime with the language codes extracted from the LC_MESSAGES environment variable. It will also ensure that these codes are not included twice in the list. If LC_MESSAGES is set to "C" only the Translation−en file (if available) will be used. To force APT to use no Translation file use the setting Acquire::Languages=none. "none" is another special meaning code which will stop the search for a suitable Translation file. This tells APT to download these translations too, without actually using them unless the environment specifies the languages. So the following example configuration will result in the order "en, de" in an English locale or "de, en" in a German one. Note that "fr" is downloaded, but not used unless APT is used in a French locale (where the order would be "fr, de, en").

Acquire::Languages { "environment"; "de"; "en"; "none"; "fr"; };

Note: To prevent problems resulting from APT being executed in different environments (e.g. by different users or by other programs) all Translation files which are found in /var/lib/apt/lists/ will be added to the end of the list (after an implicit "none").

ForceIPv4
When downloading, force to use only the IPv4 protocol.

ForceIPv6
When downloading, force to use only the IPv6 protocol.

MaxReleaseFileSize
The maximum file size of Release/Release.gpg/InRelease files. The default is 10MB.

EnableSrvRecords
This option controls if apt will use the DNS SRV server record as specified in RFC 2782 to select an alternative server to connect to. The default is "true".

AllowInsecureRepositories
Allow update operations to load data files from repositories without sufficient security information. The default value is "false". Concept, implications as well as alternatives are detailed in **apt-secure**(8).

AllowWeakRepositories
Allow update operations to load data files from repositories which provide security information, but these are deemed no longer cryptographically strong enough. The default value is "false". Concept, implications as well as alternatives are detailed in **apt-secure**(8).

AllowDowngradeToInsecureRepositories
Allow that a repository that was previously gpg signed to become unsigned during an update operation. When there is no valid signature for a previously trusted repository apt will refuse the update. This option can be used to override this protection. You almost certainly never want to enable this. The default is false. Concept, implications as well as alternatives are detailed in **apt-secure**(8).

Changelogs::URI scope
Acquiring changelogs can only be done if an URI is known from where to get them. Preferable the Release file indicates this in a 'Changelogs' field. If this isn't available the Label/Origin field of the Release file is used to check if a Acquire::Changelogs::URI::Label::*LABEL* or Acquire::Changelogs::URI::Origin::*ORIGIN* option exists and if so this value is taken. The value in the Release file can be overridden with Acquire::Changelogs::URI::Override::Label::*LABEL* or Acquire::Changelogs::URI::Override::Origin::*ORIGIN*. The value should be a normal URI to a text file, except that package specific data is replaced with the placeholder @CHANGEPATH@. The value for it is: 1. if the package is from a component (e.g. main) this is the first part otherwise it is omitted, 2. the first letter of source package name, except if the source package name starts with 'lib' in which case it will be the first four letters. 3. The complete source package name. 4. the complete name again and 5. the source version. The first (if present), second, third and fourth part are separated by a slash ('/') and between the fourth and fifth part is an underscore ('_'). The special value 'no' is available for this option indicating that this source can't be used to acquire changelog files from. Another source

will be tried if available in this case.

BINARY SPECIFIC CONFIGURATION

Especially with the introduction of the **apt** binary it can be useful to set certain options only for a specific binary as even options which look like they would effect only a certain binary like **APT::Get::Show−Versions** effect **apt−get** as well as **apt**.

Setting an option for a specific binary only can be achieved by setting the option inside the **Binary::***specific−binary* scope. Setting the option **APT::Get::Show−Versions** for the **apt** only can e.g. by done by setting **Binary::apt::APT::Get::Show−Versions** instead.

Note that as seen in the DESCRIPTION section further above you can't set binary−specific options on the commandline itself nor in configuration files loaded via the commandline.

DIRECTORIES

The Dir::State section has directories that pertain to local state information. lists is the directory to place downloaded package lists in and status is the name of the **dpkg**(1) status file. preferences is the name of the APT preferences file. Dir::State contains the default directory to prefix on all sub−items if they do not start with / or ./.

Dir::Cache contains locations pertaining to local cache information, such as the two package caches srcpkgcache and pkgcache as well as the location to place downloaded archives, Dir::Cache::archives. Generation of caches can be turned off by setting pkgcache or srcpkgcache to "". This will slow down startup but save disk space. It is probably preferable to turn off the pkgcache rather than the srcpkgcache. Like Dir::State the default directory is contained in Dir::Cache

Dir::Etc contains the location of configuration files, sourcelist gives the location of the sourcelist and main is the default configuration file (setting has no effect, unless it is done from the config file specified by **APT_CONFIG**).

The Dir::Parts setting reads in all the config fragments in lexical order from the directory specified. After this is done then the main config file is loaded.

Binary programs are pointed to by Dir::Bin. Dir::Bin::Methods specifies the location of the method handlers and gzip, bzip2, lzma, dpkg, apt−get dpkg−source dpkg−buildpackage and apt−cache specify the location of the respective programs.

The configuration item RootDir has a special meaning. If set, all paths will be relative to RootDir, *even paths that are specified absolutely*. So, for instance, if RootDir is set to /tmp/staging and Dir::State::status is set to /var/lib/dpkg/status, then the status file will be looked up in /tmp/staging/var/lib/dpkg/status. If you want to prefix only relative paths, set Dir instead.

The Ignore−Files−Silently list can be used to specify which files APT should silently ignore while parsing the files in the fragment directories. Per default a file which ends with .disabled, ~, .bak or .dpkg−[a−z]+ is silently ignored. As seen in the last default value these patterns can use regular expression syntax.

APT IN DSELECT

When APT is used as a **dselect**(1) method several configuration directives control the default behavior. These are in the DSelect section.

Clean
: Cache Clean mode; this value may be one of always, prompt, auto, pre−auto and never. always and prompt will remove all packages from the cache after upgrading, prompt (the default) does so conditionally. auto removes only those packages which are no longer downloadable (replaced with a new version for instance). pre−auto performs this action before downloading new packages.

options
: The contents of this variable are passed to **apt-get**(8) as command line options when it is run for the install phase.

Updateoptions
: The contents of this variable are passed to **apt-get**(8) as command line options when it is run for the update phase.

PromptAfterUpdate
If true the [U]pdate operation in **dselect**(1) will always prompt to continue. The default is to prompt only on error.

HOW APT CALLS DPKG(1)

Several configuration directives control how APT invokes **dpkg**(1). These are in the DPkg section.

options
This is a list of options to pass to **dpkg**(1). The options must be specified using the list notation and each list item is passed as a single argument to **dpkg**(1).

Path
This is a string that defines the **PATH** environment variable used when running dpkg. It may be set to any valid value of that environment variable; or the empty string, in which case the variable is not changed.

Pre−Invoke, **Post−Invoke**
This is a list of shell commands to run before/after invoking **dpkg**(1). Like options this must be specified in list notation. The commands are invoked in order using /bin/sh; should any fail APT will abort.

Pre−Install−Pkgs
This is a list of shell commands to run before invoking **dpkg**(1). Like options this must be specified in list notation. The commands are invoked in order using /bin/sh; should any fail APT will abort. APT will pass the filenames of all .deb files it is going to install to the commands, one per line on the requested file descriptor, defaulting to standard input.

Version 2 of this protocol sends more information through the requested file descriptor: a line with the text VERSION 2, the APT configuration space, and a list of package actions with filename and version information.

Each configuration directive line has the form key=value. Special characters (equal signs, newlines, nonprintable characters, quotation marks, and percent signs in key and newlines, nonprintable characters, and percent signs in value) are %−encoded. Lists are represented by multiple key::=value lines with the same key. The configuration section ends with a blank line.

Package action lines consist of five fields in Version 2: package name (without architecture qualification even if foreign), old version, direction of version change (< for upgrades, > for downgrades, = for no change), new version, action. The version fields are "−" for no version at all (for example when installing a package for the first time; no version is treated as earlier than any real version, so that is an upgrade, indicated as − < 1.23.4). The action field is "**CONFIGURE**" if the package is being configured, "**REMOVE**" if it is being removed, or the filename of a .deb file if it is being unpacked.

In Version 3 after each version field follows the architecture of this version, which is "−" if there is no version, and a field showing the MultiArch type "same", "foreign", "allowed" or "none". Note that "none" is an incorrect typename which is just kept to remain compatible, it should be read as "no" and users are encouraged to support both.

The version of the protocol to be used for the command *cmd* can be chosen by setting DPkg::Tools::options::*cmd*::Version accordingly, the default being version 1. If APT isn't supporting the requested version it will send the information in the highest version it has support for instead.

The file descriptor to be used to send the information can be requested with DPkg::Tools::options::*cmd*::InfoFD which defaults to 0 for standard input and is available since version 0.9.11. Support for the option can be detected by looking for the environment variable **APT_HOOK_INFO_FD** which contains the number of the used file descriptor as a confirmation.

Run−Directory
APT chdirs to this directory before invoking **dpkg**(1), the default is **/**.

Build−options
These options are passed to **dpkg-buildpackage**(1) when compiling packages; the default is to disable signing and produce all binaries.

DPkg::ConfigurePending
If this option is set APT will call **dpkg −−configure −−pending** to let **dpkg**(1) handle all required configurations and triggers. This option is activated by default, but deactivating it could be useful if you want to run APT multiple times in a row − e.g. in an installer. In this scenario you could deactivate this option in all but the last run.

PERIODIC AND ARCHIVES OPTIONS

APT::Periodic and APT::Archives groups of options configure behavior of apt periodic updates, which is done by the /usr/lib/apt/apt.systemd.daily script. See the top of this script for the brief documentation of these options.

DEBUG OPTIONS

Enabling options in the Debug:: section will cause debugging information to be sent to the standard error stream of the program utilizing the apt libraries, or enable special program modes that are primarily useful for debugging the behavior of apt. Most of these options are not interesting to a normal user, but a few may be:

- Debug::pkgProblemResolver enables output about the decisions made by dist−upgrade, upgrade, install, remove, purge.
- Debug::NoLocking disables all file locking. This can be used to run some operations (for instance, apt−get −s install) as a non−root user.
- Debug::pkgDPkgPM prints out the actual command line each time that apt invokes **dpkg**(1).
- Debug::IdentCdrom disables the inclusion of statfs data in CD−ROM IDs.

A full list of debugging options to apt follows.

Debug::Acquire::cdrom
Print information related to accessing cdrom:// sources.

Debug::Acquire::ftp
Print information related to downloading packages using FTP.

Debug::Acquire::http
Print information related to downloading packages using HTTP.

Debug::Acquire::https
Print information related to downloading packages using HTTPS.

Debug::Acquire::gpgv
Print information related to verifying cryptographic signatures using gpg.

Debug::aptcdrom
Output information about the process of accessing collections of packages stored on CD−ROMs.

Debug::BuildDeps
Describes the process of resolving build−dependencies in **apt-get**(8).

Debug::Hashes
Output each cryptographic hash that is generated by the apt libraries.

Debug::IdentCDROM
Do not include information from statfs, namely the number of used and free blocks on the CD−ROM filesystem, when generating an ID for a CD−ROM.

Debug::NoLocking
Disable all file locking. For instance, this will allow two instances of “apt−get update” to run at the

same time.

Debug::pkgAcquire
Log when items are added to or removed from the global download queue.

Debug::pkgAcquire::Auth
Output status messages and errors related to verifying checksums and cryptographic signatures of downloaded files.

Debug::pkgAcquire::Diffs
Output information about downloading and applying package index list diffs, and errors relating to package index list diffs.

Debug::pkgAcquire::RRed
Output information related to patching apt package lists when downloading index diffs instead of full indices.

Debug::pkgAcquire::Worker
Log all interactions with the sub−processes that actually perform downloads.

Debug::pkgAutoRemove
Log events related to the automatically−installed status of packages and to the removal of unused packages.

Debug::pkgDepCache::AutoInstall
Generate debug messages describing which packages are being automatically installed to resolve dependencies. This corresponds to the initial auto−install pass performed in, e.g., apt−get install, and not to the full apt dependency resolver; see Debug::pkgProblemResolver for that.

Debug::pkgDepCache::Marker
Generate debug messages describing which packages are marked as keep/install/remove while the ProblemResolver does his work. Each addition or deletion may trigger additional actions; they are shown indented two additional spaces under the original entry. The format for each line is MarkKeep, MarkDelete or MarkInstall followed by package−name <a.b.c −> d.e.f | x.y.z> (section) where a.b.c is the current version of the package, d.e.f is the version considered for installation and x.y.z is a newer version, but not considered for installation (because of a low pin score). The later two can be omitted if there is none or if it is the same as the installed version. section is the name of the section the package appears in.

Debug::pkgDPkgPM
When invoking **dpkg**(1), output the precise command line with which it is being invoked, with arguments separated by a single space character.

Debug::pkgDPkgProgressReporting
Output all the data received from **dpkg**(1) on the status file descriptor and any errors encountered while parsing it.

Debug::pkgOrderList
Generate a trace of the algorithm that decides the order in which apt should pass packages to **dpkg**(1).

Debug::pkgPackageManager
Output status messages tracing the steps performed when invoking **dpkg**(1).

Debug::pkgPolicy
Output the priority of each package list on startup.

Debug::pkgProblemResolver
Trace the execution of the dependency resolver (this applies only to what happens when a complex dependency problem is encountered).

Debug::pkgProblemResolver::ShowScores
Display a list of all installed packages with their calculated score used by the pkgProblemResolver. The description of the package is the same as described in Debug::pkgDepCache::Marker

Debug::sourceList
Print information about the vendors read from /etc/apt/vendors.list.

Debug::RunScripts
Display the external commands that are called by apt hooks. This includes e.g. the config options DPkg::{Pre,Post}−Invoke or APT::Update::{Pre,Post}−Invoke.

EXAMPLES

/usr/share/doc/apt/examples/configure−index.gz is a configuration file showing example values for all possible options.

FILES

/etc/apt/apt.conf
APT configuration file. Configuration Item: Dir::Etc::Main.

/etc/apt/apt.conf.d/
APT configuration file fragments. Configuration Item: Dir::Etc::Parts.

SEE ALSO

apt-cache(8), **apt-config**(8), **apt_preferences**(5).

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

Daniel Burrows <dburrows@debian.org>
Initial documentation of Debug::*.

NOTES

1. APT bug page
http://bugs.debian.org/src:apt