

# Apt-Cacher-NG User Manual

---

Apt-Cacher NG is a caching proxy for software packages which are downloaded by Unix/Linux system distribution mechanisms from mirror servers accessible via HTTP.

This manual provides an overview of Apt-Cacher-NG's features and a walk through the required configuration steps for server administrators and users of the proxy.

# Contents

---

Chapter 1: Introduction	5
Chapter 2: Running apt-cacher-ng	6
Chapter 3: Basic Configuration	7
3.1 Server Configuration	7
3.2 Client Configuration	7
Chapter 4: Advanced Server Configuration	9
4.1 Vocabulary	9
4.2 Configuration file types	9
4.3 Repositories and URL mapping	10
4.3.1 Writing Remap-... configuration	11
4.3.2 Special tricks and additional notes	12
Chapter 5: Security	14
5.1 Access control by IP interface	14
5.2 Access control with libwrap	14
5.3 Access control with inetd	15
5.4 Access control with iptables	15
5.5 Target port filter	15
Chapter 6: Distribution specific instructions	16
6.1 Debian and Ubuntu	16
6.2 OpenSUSE	16
6.3 Fedora Core	16
6.4 Arch Linux	16
6.5 Sourceforge mirror network	16
6.6 Cygwin mirrors	17

6.7 Limited expiration . . . . .	17
Chapter 7: Maintenance . . . . .	18
7.1 Cache cleanup . . . . .	18
7.1.1 Manual expiration . . . . .	18
7.1.2 Automated cache cleanup . . . . .	19
7.1.3 Keeping latest versions of expired package files . . . . .	20
7.2 Removal of distribution releases . . . . .	20
Chapter 8: HOWTOs and FAQ . . . . .	22
8.1 Package import . . . . .	22
8.2 Cache overview . . . . .	23
8.3 Access to SSL/TLS remotes (HTTPS) . . . . .	24
8.4 JIGDO usage . . . . .	25
8.5 Avoid use of apt-cacher-ng for certain hosts . . . . .	25
8.6 Avoid caching for certain domains or certain file types . . . . .	25
8.7 How to make big download series faster . . . . .	26
8.8 How to import DVDs or ISO images . . . . .	26
8.9 How to integrate DVDs or ISO image data . . . . .	26
8.10 How to execute commands before and after going online? . . . . .	26
8.11 Listen to only specific interfaces or IP protocols . . . . .	27
8.12 How to avoid use of IPv4 (or IPv6) where possible? . . . . .	27
8.13 Use the proxy without storing all data twice . . . . .	27
8.14 Partial Mirroring . . . . .	28
Chapter 9: Troubleshooting . . . . .	29
9.1 Debugging . . . . .	29
9.2 Problem: keeps delivering damaged files . . . . .	29
9.3 Problem: regular expiration action reproducibly aborts . . . . .	30
9.4 Problem: cacher suddenly terminates, log reports IO errors . . . . .	30
9.5 Problem: download fails with 503 ... status message . . . . .	30
9.6 Problem: apt-get freezes when downloading files . . . . .	31
9.7 apt-get reports corrupted bzip2 data . . . . .	31

9.8 Problem: apt-cacher-ng refuses to start with "Address already in use" . . . . .	32
Chapter 10: Known Bugs and Limitations . . . . .	33
Chapter 11: Contact . . . . .	34

# Chapter 1: Introduction

---

apt-cacher-ng attempts to achieve the same goals as related proxies - it acts as a proxy which is used by clients in the local network to share the data that has been downloaded. It monitors the state of packages and is capable of merging downloads of the same packages from different locations (real or simulated).

The program reuses many ideas behind the other famous proxy, its predecessor apt-cacher 1.x (which has been written in Perl). In contrast to apt-cacher, different aspects have been declared as primary targets during the development of apt-cacher-ng:

- lightweight implementation - allow use on systems with low memory and processing resources
- internal (native) threading - avoiding process fork'ing wherever possible, avoiding kludges for pseudo-thread synchronization, avoiding relying on special file system features for internal operations where possible
- real (effective) support of HTTP pipelining, using an internal client with native stream control (having the nice side effect: reduction of resource overhead and minimization of possible points of failure)
- avoiding featuritis where they cause too much bloat and the functionality can be provided by native OS features
- reliable but efficient content merging in the local package pool, avoiding delivering of wrong data.

As with apt-cacher, explicit tracking of dynamically changed and unchanged files is established, and the use in non-Debian environment is supported.

Long story: Not all goals have been achieved. The initial plan of using background databases to merge any download from any arbitrary location has been dropped because of complexity and performance considerations, reliable heuristics could not be found either. Instead, a semi-automated solution has been created which used machine-parsable files with mirror information, like the one available for Debian mirrors in Debian's CVS repository.

## Chapter 2: Running apt-cacher-ng

---

Run "build/apt-cacher-ng -c conf" when configured where conf is the configuration directory. See section 4.2 for details on possible and required contents of this directory.

Most options from the configuration file can also be passed through command line parameters. Just append them with the same format as in the configuration file but without separating spaces inside, e.g.

```
Port:4855 ForeGround=1
```

For convenience, the colon can also be replaced with the equals sign and letter case does not matter, so this is also possible:

```
port=4855 foreground:1.
```

## Chapter 3: Basic Configuration

---

### 3.1 Server Configuration

Unlike some rumors on the internet claim, there should be no need for exhausting configuration work to just test apt-cacher-ng and run it with default parameters. It's actually designed to bootstrap most of its working environment without additional help.

The package setup scripts used by distributions should already prepare working initial settings for apt-cacher-ng. Check the file `/etc/apt-cacher-ng/acng.conf` file where most settings are explained. For the beginning they should not be changed, the only interesting setting present there is the TCP port. See Advanced Server Configuration for details.

There is also a daily cron job which executes some maintenance work. Additional automated control commands can be added by administrator.

### 3.2 Client Configuration

From the client side, apt-cacher-ng can be used as a drop-in replacement for apt-cacher. The same rules apply, e.g. Debian/Ubuntu users should EITHER:

- Specify the caching machine as HTTP Proxy for your download client. This can be usually done (temporarily) with the environment variable `http_proxy` or (permanently) in some configuration file or dialog. For APT, this means putting a line like the following into a file like `/etc/apt/apt.conf.d/02proxy`:

```
Acquire::http::proxy "http://CacheServerIp:3142";
```

OR:

- Replace all mirror hostnames with cachinghost/hostname in `sources.list`, so

```
deb http://ftp.uni-kl.de/debian etch main
```

should now become:

```
deb http://192.168.0.17:3142/ftp.uni-kl.de/debian etch main
```

*(assuming that CacheServerIp is 192.168.0.17 and the service port is 3142).*

These both methods have their own pros and cons. The last method can be used with clients which cannot configure an additional http proxy for some reason. The disadvantages are the worse progress visibility (all downloads seem to come from the same machine) and some resource usage limits might be hit (i.e. maximum number of simultaneous downloads from the "same" machine). It might also require to modify many different URLs and some of those URLs might be hardcoded in software and not be accessible to the user.

The former method is more convenient since it usually means less configuration work; however, it implies that all relevant client programs respect the proxy setting and use it from this central location.

Mixing the configuration methods is usually possible but not within the same client program instance. Doing that (going with proxy mode AND use rewritten URLs) will probably confuse the server: in best-case, the connection will be slower because of a little transport loop on the server side, and in the worst-case, the target URL will eventually become not resolvable and the downloads will just fail.

Using SSL/TLS transport (i.e. https urls) is also possible with some restrictions, see section 8.3 for details.



# Chapter 4: Advanced Server Configuration

---

## 4.1 Vocabulary

This chapter introduces some terminology which is needed to understand the functionality of apt-cacher-ng; it's recommended to understand it before continuing with the advanced configuration.

- "Backend": a text file consisting of a list of mirror URLs, one per line (a more complex RFC822-like format is also supported). Used for URL remapping; see section 4.3.
- "Volatile files": nothing to do with debian-volatile, volatile here only means that they are volatile, i.e. their contents are expected to be regularly changed on the server. For example, metadata pertaining to package files stored in a remote archive is classified as 'volatile'. They are usually 'index files' known as Packages, Sources, Release, Pdiff and the like.
- "Package files": files that contain software packages and other "solid" data: DEBs, source files for their creation (.tar.gz, .diff, .dsc), various metadata which is not subject to change after first appearance on the server.
- "Configuration line": one single line in the configuration file. Some examples in this chapter may contain wrapped lines but should be stored as a single line in the configuration.

## 4.2 Configuration file types

By default, the /etc/apt-cacher-ng directory (or the one specified with program options) contains all config files, HTML page templates, the stylesheet and other text-based support files used by apt-cacher-ng. The contents may vary depending on the installation of apt-cacher-ng, refer to the package documentation for Linux Distribution packages.

There are a few certain file types distinguished by apt-cacher-ng:

1. Main configuration files:

\*.conf files are assumed to contain configuration directives in the form of "key: value" pairs. The package comes with a commented example configuration file. apt-cacher-ng reads all files matching \*.conf in alphabetical order and merges the contents. For options documentation, see commented example file shipped with apt-cacher-ng (conf/ directory in original source).

For security reasons, the files can be made readable only to the daemon and administrator accounts, e.g. when they contain passwords or other sensitive data.

2. URL lists and remote repository list files. The file names are arbitrary, no special suffix is required. They are read and included during processing of configuration files and can contain data in one of the following formats:

- simple text files with one URL per line (the URL should point to the base directory of the repository, e.g. "http://ftp.de.debian.org/debian/"). A URL must start with http:// and should end with a slash
  - an RFC822-like format, with lines like 'Site: <hostname>' and 'Archive-http: /base/directory/of/repository/'. Optional fields are also used in this remapping descriptions to add more possible variants (Alias, Aliases, X-Archive-http:) of the URLs to the lookup list
3. Various support files used for the configuration web interface, named like \*.css and \*.html.
  4. \*.default files are used in some rare cases as replacement for list files having the same name without .default suffix.
  5. \*.hooks files specify custom actions which can be executed upon connection/disconnection (see section 4.3.2 for details).

Except from .conf files, most files listed above can be moved to another "support" directory and the daemon will look for them therein if they are not present in the primary configuration directory. This feature is intended to help keeping pure configuration data and mostly static data in different locations. The directory path is specified at build time and can be overridden with the *SupportDir* directive (and if used, this should be set as early as possible).

## 4.3 Repositories and URL mapping

With the most simple configuration, apt-cacher-ng will act almost like an ordinary HTTP proxy with improved caching behaviour. When files are requested, they are downloaded from a remote location specified in client's request and are stored in a unique way.

However, for some use cases it can be beneficial to specify additional rules to achieve further improvements, e.g. in order to detect and prevent avoidable downloads, to reduce space requirements for the cache directory or simply hide real download locations from the APT clients.

These modifications are generally achieved by two strategies, *Merging* and *Redirection*, which are configured in a context of a specified cache *Repository*. The configuration for them is created using one or multiple *Remap-...* configuration directives (see below).

*Merging:*

"Merging" of incoming requests can be done if some subdirectories of different remote servers are considered equal where the last part of the remote file path leads to the same file content. When specified, the internal cache content is shared and the live download stream is shared. The configuration work consists of setting an "equality list" containing a set of URLs representing the base directories (like `http://ftp.debian.org/debian` and `http://ftp.uni-kl.de/pub/linux/debian`).

*Redirection:*

With redirection, client requests cause a download from a remote location which is different from what clients requested and believe to receive from. Redirection is an optional feature; if used, it's configured by one or multiple URL(s) pointing to target servers. The URL(s) must include a directory spec which matches the directory level of the URLs in the *Merging* URL(s),

for example all ending with /ubuntu/ for usual Ubuntu mirror URLs. If redirection is not used (i.e. the target URL list is empty) the original URL from client's request is used to get the data.

*Repository:*

A (cache) repository is the internal identifier which declares the scope in which Merging/Redirection specs are applied. It also represents the name of an internal cache subdirectory.

### 4.3.1 Writing Remap-... configuration

When use cases for merging/redirection are identified and a repository name is chosen, these components are written into configuration directives starting with Remap- which follow the simple syntax:

```
Remap-RepositoryName: MergingURLs ; TargetURLs ; OptionalFlags
```

The repository name is a symbolic name which should be chosen carefully and should not be changed afterwards, otherwise the data might become inaccessible for clients until the files are extracted and reimported semi-manually. Internally, this string shares the namespace with host names and/or top directory names of other URLs. Name collisions can cause nasty side effects and should be avoided. Recommended names are made up from alphanumeric or URL-friendly characters. Also, a repository name should not be associated to a real hostname. Examples for good names: archlinux-repo, debianlocal. Examples for bad names: fedora.example.com, \_very&weird.

The TargetURLs part is optional (see Redirection description above). If multiple targets are specified, the order of servers here defines their order of preference (see also the NetworkTimeout option and additional notes below).

Both URL lists simply contain URLs separated by spaces. The strings must be properly URL-encoded. Since all URLs are assumed to belong to http:// protocol and point to a remote directory, the http:// protocol prefix and trailing slashes are optional. There is no hard limit to the number of URLs. However, for readability reasons it's recommended to put them into separate list files (see section 4.2) and specify the particular list files with tags like file:urlsDebian.list instead of writing them into a single line. Raw URLs and file:... lists can be mixed.

Fully configured Remap lines can look like:

*Example I:*

```
Remap-  
debrep: ftp.de.debian.org/debian http://ftp.at.debian.org/debian
```

for the use case: small home network, clients have de... or at... servers in their sources.list files and use acng as HTTP proxy. Now the files are still downloaded from at... or de... mirrors depending on the user request, but already cached data is served to both, at... and de... users.

*Example II:*

```
Remap-  
ubuntu: file:ubumir.lst ; 192.168.17.23/pu ca.archive.ubuntu.com/ubuntu
```

for the use case: small home network, clients have various Ubuntu mirrors (which are listed in

ubumir.lst) in their sources.list files and use acng as HTTP proxy. All requests are redirected to a mirror in the /pu directory of some local machine. When that machine is down, Canadian public server is used instead.

### 4.3.2 Special tricks and additional notes

There are some implementation details (partially explained above) and some configuration options related to repository settings which should be mentioned explicitly.

The internal cache directory tree follows the URL requests from the clients unless modified by Remapping rules. For proxy-style configuration on the user side, it is always the hostname of the requested URL. But if clients access the apt-cacher-ng server like a regular mirror (not using APT's proxy config) then it's just passed as regular directory name. And at this point, it's possible to use Remapping constructs to access random remote locations while the client assumes to download from a subdirectory of apt-cacher-ng (as http server). This is configured by simply using /some/directory/string/ instead of URLs in the Merging list to let your clients download from http://acngserver/some/directory/string/... paths.

If multiple Remap- lines for the same Repository are specified, the contents of both URL lists are merged.

On some restricted networks, it may be needed to enforce the use of predefined mirrors. If the ForceManaged option is set, only requests to URL matched in some Remap-... config is allowed.

Sometimes, it may be needed to execute a system command before connection to certain machines is established. This is possible by associating commands with a repository declaration, i.e. by storing a file named like repositoryname.hooks in the main configuration directory. It can contain PreUp, Down and DownTimeout settings. PreUp/Down are executed by the system shell and it's up to the administrator to make sure that no malicious code is contained there and that the execution of these commands does not cause significant delays for other apt-cacher-ng users. See package documentation for an exemplary hooks file.

If the Redirection part contains multiple URLs, the server prefers to use them in the order of appearance. On success, the first target is used all the time, and so this should be the preferred mirror (note: "success" means getting a started download or a non-critical failure in this context. A "404 File not found" status is not considered critical since client's apt can expect and use it to check the existence of remote files and then change its own behaviour accordingly).

And finally, there is an optional third field in the Remap directives which can contain extra flags to modify downloading behavior in the scope of that particular cache repository.

- `keyfile=...` The meaning of this setting is: if any real download error (status code 400 and higher) happens on a file which path ends with the specified string then the target server is blacklisted (considered faulty) immediately and this download (and subsequent ones requested by this client connection) are retried from other servers (see TargetURLs description above). Can be used multiple times to define a list. See below for documented example.
- `deltasrc=URL` Configures the base URL used to download .debdelta files. The path hierarchy below this URL should correspond to the source URLs and file paths in the cache. Only one URL can be specified at the moment. It is used for explicit mirroring operations, see section 8.14 for details.

- `proxy=proxyspec` Configures an alternative proxy URL which overrides the global proxy setting in the context of this repository. Can be set empty to disable proxy usage.

Config example:

```
Remap-debrep: file:deb_mirror*.gz ; file:backends_debian ;  
             keyfile=Release keyfile=.deb
```

If the first mirror from `backends_debian` goes wild and returns 404 responses for everything then the next candidate will be used. However, while this feature can improve redundancy for certain installations it needs to be used with care! Some file types are allowed to be missing and `apt` interprets their absence to change its behavior as needed. `keyfile=` should only match files which have an essential role and which disappearance is undoubtful indication of a broken server.

## Chapter 5: Security

---

Like many data storing daemons with predictable filenames, apt-cacher-ng is vulnerable to symlink attacks and similar malicious actions. Therefore, the user must make sure that the cache and log directories are writable only to the user account under which apt-cacher-ng is running.

General network access control can be established with operating system's facilities in kernel space (see below) or in user space (e.g. with inetd, see section 5.3). There is also experimental built-in access filter feature which uses the TCP Wrapper library for filtering (i.e. uses the same mechanism as tcpd). See section 5.2 for details.

As to the program's internal security, apt-cacher-ng has been developed with concern about a certain grade of attacks from internal users as well as from malicious external hosts. However, no guarantees can be made about the security of the program. It is recommended to run apt-cacher-ng under a dedicated system account which has no write access to any files outside of the cache and log directories. Refer to the manuals of the administration utilities of the system distribution (like start-stop-daemon) to create the required configuration.

If relaxed permissions are required, e.g. to make files group-writable, this can be established through the appropriate use of umask command in the startup scripts of apt-cacher-ng (see `/etc/default/apt-cacher-ng`, for example) and the sticky bit on the cache directories (see `chmod(1)` manpage for details). However, write permissions should be assigned very carefully because that may make the server vulnerable to symlink attacks and like-minded activities.

The management web interface which allows execution of certain operations can be protected by HTTP credentials (username/password). The trust levels for most actions depend on their purpose. Those which should never be able to destroy important local files can be triggered by anyone if no password is set. And on the other hand, some operations are considered dangerous and can only be accessed when the admin password is configured and was entered correctly.

### 5.1 Access control by IP interface

A simple control method for incoming connections is listening only to network interfaces which are inside a secure perimeter, e.g. belong to the local network. See section 8.11 for details on this configuration parameter. The default setting is listening to all interfaces.

### 5.2 Access control with libwrap

If the access control with operating system's ip filters is not possible, the embedded filtering might be using instead. It is turned on if the feature is included at build time and the `UseWrap` option is set to non-zero. If `UseWrap` is not set at all, it might be enabled implicitly if the `hosts.allow` or `hosts.deny` files have rules for apt-cacher-ng.

For more information about host access list creation, refer to `hosts_access(5)` manual page (or manual pages for `hosts.allow` and `hosts.deny`).

The libwrap filtering method has a little drawback compared to alternatives. The host checks are called in single-threaded context and can add some lag for other incoming connections.

## 5.3 Access control with inetd

In some situations, access filtering by client IP/hostname might be not supported directly or there are other reasons to use inetd to wrap access to apt-cacher inetd. For this case, an inetd daemon is shipped with the package which makes the use of tcpd possible. Installation is done in following steps:

1. compile the inetd bridge tool "in.acng", if not already done (check `/usr/lib/apt-cacher-ng`).
2. Edit apt-cacher-ng's configuration (acng.conf, for example), and set a path for a new file in a writable directory, like this:

```
SocketPath: /var/run/apt-cacher-ng/socket
```

3. Edit /etc/inetd.conf and add following line with appropriate path names and TCP port:

```
3143  stream  tcp  nowait  user  /usr/sbin/tcpd  
      /usr/local/sbin/in.acng  /var/run/apt-cacher-ng/socket
```

4. Edit hosts.allow and other files to configure ACLs for port 3143. See tcpd(8) and related manpages for further details.
5. Configure clients to use the alternative port (3143 in the example above).

## 5.4 Access control with iptables

Looking at performance, the most efficient way to establish access control by source IP is the use of system's native mechanisms. For the Linux kernel, it can be created with iptables rules like in the following example, usually stored in system's firewall setup script (or any init script like `/etc/rc.local`):

```
iptables -A INPUT -p tcp --dport 3142 --source 127.0.0.0/8 -j ACCEPT  
iptables -A INPUT -p tcp --dport 3142 --source 192.168.0.0/16 -j ACCEPT  
iptables -A INPUT -p tcp --dport 3142 -j REJECT --reject-with tcp-reset
```

## 5.5 Target port filter

In the default configuration, *apt-cacher-ng* restricts the choice of the target port on the remote mirror to the default HTTP port (80). This is intended to prevent creation of HTTP tunnels and similar malicious user activities.

Cache administrator can define an own list of allowed ports by setting them as (space separated) list with the AllowUserPorts configuration directive.

Alternatively, access to custom TCP ports can be established through custom remapping specification.

## Chapter 6: Distribution specific instructions

---

### 6.1 Debian and Ubuntu

Use as is. Report bugs using reportbug (Debian) or to Launchpad (Ubuntu).

### 6.2 OpenSUSE

Server can be used as is with limited expiration (see below, and see `INSTALL` file for compilation hints). The merging mode (multiple servers mapped into the same repository) is not preconfigured in the example configuration. This is object to research, competent support is required.

Clients can configure apt-cacher-ng as central proxy in Yast ("Network devices"/"Proxy" tab). If this is not desirable then each software source can be edited to be redirected through the server. This can be done in the Software Installer view of Yast2, see menu Configuration/Repositories. To edit them quickly, switch to URL editing mode and insert `host:port/` (of the apt-cacher-ng server) between `http://` and the source server name.

### 6.3 Fedora Core

Attempts to add apt-cacher-ng support ended up in pain and the author lost any motivation in further research on this subject.

### 6.4 Arch Linux

Mostly usable. The mirror structure design has been identified by trial-and-error and the regular expressions might need some tuning by experts.

The installer seems to have no way to specify a dedicated proxy but it's possible to edit the source URL and insert the `host:port/` part into it. The pacman mirror list can be modified the same way.

Expiration code should work in the basic mode (index data is extracted from `*.db.tar.gz` files). File checksum checking mode might also work (untested). The example configuration contains a preconfigured list of mirrors which can be rebuilt with the Makefile if needed. The preferred backend server can be specified like with other distributions (see above for details).

### 6.5 Sourceforge mirror network

Not a Linux distro but commonly used by those to download certain files. Therefore most mirrors can get unified access cache sharing the files in the same cache repository. Some `.exe` files there are never expired.



## 6.6 Cygwin mirrors

While not being a pure Linux distro it's mostly GNU and has a nice mirror setup. Proxy server can be used as is with limited expiration (see below). Apt-cacher-ng can also be compiled and used on Windows machines in Cygwin environment (see `INSTALL` file for details).

Clients need to specify the server as HTTP proxy in the `setup.exe` wizard, only HTTP mirrors should be selected in the mirror list.

## 6.7 Limited expiration

The expiration code for Non-Debian/Ubuntu repositories is quite limited due to lack of manpower or know-how. For some distros, the support is limited to checks of the filename and no further validation is supported. Therefore, the extra validation of path location or file contents should NOT be turned on when running expiration with data from that distros in the cache, because good data may be deleted in this case.

# Chapter 7: Maintenance

---

There are few optional tasks that need to be executed by the administrator from time to time or during the initial configuration.

## 7.1 Cache cleanup

If a package is no longer downloadable by APT clients then its files are also not referenced in any index file and can be removed. This rule also applies to most volatile files from the distribution metadata. For example, Debian's Release file references some Packages and Sources files or Diff-Index file, and those do reference most other non-volatile files (binary packages, source packages, index diffs, ...).

### 7.1.1 Manual expiration

To run the cleanup action manually visit the report page in a browser and trigger the Expiration operation there.

There are different flags configuring the parameters of this tracking described below. Usually just the filename is sufficient to consider a file in the cache as a valid (downloadable) file. This is ok in most cases but sometimes leads to false positives, i.e. when another repository in the cache refers to a file with the same name but the reference to the original location is gone. On the other hand there can be cases where the assignment to different repositories happened by mistake and administrator would like to merge repositories later on.

For most files the checksum values are also provided in the index files and so the file contents can be validated as well. This requires reading of the whole cache archive to generate local checksums. It should also not be done when apt-cacher-ng is being used (file locking is not used here).

Usually it's necessary to bring various index files (Release,Sources,Packages,Index) in sync with the repository. This is necessary because apt works around the whole file download by fetching small patches for the original file, and this mode of operation is not supported yet by apt-cacher-ng (and might still be unreliable). When this synchronization fails, the index files might be incomplete or obsolete or damaged, and they might no longer contain references to some files in the cache. Abortion of the cleanup process is advisable in this case.

There is also a precaution mechanism designed to prevent the destruction of cache contents when some volatile index files have been lost temporarily. The results of cache examination are stored in a list with the date when the particular files became orphaned. The removals are only executed after few days (configurable, see configuration file) unless they are removed from this list in the meantime.

Parameters of Expiration:

### *Skip header checks*

By default, header description file of every package is opened and checked for bad data and for obvious inconsistencies (like local file being larger than specified by server). Which means opening reading a few kilobytes from disk for almost every file in the cache, and slightly degrades performance of the process. This option skips that basic checks.

### *Stop cleanup on errors during index update step*

Index files update is done first, on errors the expiration will be interrupted.

### *Validate by file name AND file directory*

This option can be used to remove distribution stages. Example: to remove "oldstable" one just needs to delete the "Release" files in the cache and run *Expiration* with this option two times. There are some issues with this mode operation, see above for details.

### *Validate by file name AND file contents (through checksum)*

Checking file size and contents where possible against the metadata in the index files. Note: future calls of Expiration process without this option will discard the results of this check and forget about corrupted files. Therefore, an action on this files needs to be done ASAP, like truncating them (see below) or removing via the removal operation (using the checkbox and the Delete button, see process output) or via the "Delete all unreferenced files" operation on the main control page.

### *Force the download of index files*

Sometimes it may be needed to redownload all index files, explicitly replacing the cached versions. This flag enables this behaviour.

### *Purge unreferenced files after scan*

Avoid the use of the orphan list and delete files instead. This option is dangerous and should not be used unless when absolutely no mistakes/problems can happen. Instead, it's possible to view the orphan list later and delete then (see control web interface).

### *Truncate damaged files immediately*

If a file has been identified as damaged, it will be truncated (file size reset to 0). Setting this option is a good compromise for debugging purposes compared to the simple deletion since it will keep the header files on the disk, for further analysis of the problem's cause.

### *More verbosity*

Shows more information, e.g. each scanned file when used with some of the other options. This might result in a very large HTML page, making the watching HTML browser very slow.

In addition to the default scan run, there are some "Direct Action" buttons in the Web frontend. It's possible to see the temporary list of files that have been identified as orphaned (unreferenced), and it's possible to delete all files from that list immediately. To be used carefully!

## **7.1.2 Automated cache cleanup**

A script called `expire-caller.pl` is shipped with the package. This script effectively

implements a HTTP client which operates like a human would do when running the expiration manually (see above). It can also extract the operator password and unix socket file path from the local configuration file. On Debian installations it is called by the file `/etc/cron.daily/apt-cacher-ng` so it should run automatically as daily cron task. The results are usually not reported unless an error occurs, in which case some hints are written to the standard error output (i.e. sent in cron mails).

The operator script can take some options from the environment, see below. The default operation mode is calling the expiration operation with default parameters and with credentials from local system's apt-cacher-ng installation. However, this can be changed with `ACNGREQ` variable.

`DEBUG=1`

If set to non-empty and not 0, the temporary HTML output is reported to the console. For debugging purposes only.

`ACNGIP=10.0.1.3`

The network address for remote connection may be guessed incorrectly by the operator script. This variable can specify an explicit target to connect to, e.g. the same IP as the one used by the clients (unless this network connection is somehow restricted in the local setup).

`HOSTNAME=localOrPublicName`

When an error occurs, the operator script most likely adds an URL to be opened for further investigation. The host name of in this URL can be customized, i.e. can be set to a public domain name representing the server as accessible from the administrator's machine.

`ACNGREQ=cgiparameters`

Override the auto-detected command parameters with a custom set. This is the part of a command URL from the management interface after the `?` sign.

### 7.1.3 Keeping latest versions of expired package files

Sometimes it makes sense to keep a couple of versions of (Debian) packages even after they have been removed from remote source. It is possible to set an exceptional rule for package files which follow the naming and versioning scheme of `.deb`-packages. This extra handling is configured by the `KeepExtraVersions` options which tells how many of the top-latest versions shall be kept. The cache system needs the `dpkg` program and sufficient CPU power (depending on the option value).

## 7.2 Removal of distribution releases

Sometimes it's needed to remove all files from a distribution, i.e. when a new release became Stable and older package files are still lying around. In perfect conditions the reference tracking described above should take care of it and remove them soon.

However, this solution will fail if the release files are still available on the server AND apt-cacher-ng learned their real location (i.e. the code name instead of not the release state name) and so they are refreshed during regular expiration.

After all, if the old release is no longer used by local cache users then the extra disk usage

becomes a problem. This problem will go away after many months when the old release files are finally deleted on the servers, then the package expiration will start complaining for some days (the expiration delay) and only then the finally unreferenced files will be removed.

To speed up this process, the local administrator can remove the traces of the old distribution release from the archive. Either the top-level "Release" files, or even the whole index file trees relevant for certain releases.

To make this task easier, a "brutal" script called `distkill.pl` is shipped with `apt-cacher-ng`. It runs interactively, it scans the package directory and presents an overview of index file trees assumed to represent distro releases. Then it provides a command prompt to remove some immediately. The script should be used with extreme care! See section 8.2 for example of its output.

## Chapter 8: HOWTOs and FAQ

---

### 8.1 Package import

Already existing packages can be imported into apt-cacher-ng's cache pool instead of downloading them. There are some restrictions:

1. Don't try to import incomplete files. They will be refused since their contents cannot be checked against the archive metadata.
2. If possible, don't import symbolic links. Even if doing so, they should not point to other files inside of the cache and especially not to other files under the `_import` directory.

HOWTO:

1. Make sure that apt-cacher-ng has valid index files in the cache. This is the tricky part. To get them right, a client needs to download them through apt-cacher-ng once. Therefore:
  1. Configure the server and one client before doing the import. See above for instructions.
  2. Run "apt-get update" on client(s) once to teach ACNG about remote locations of (volatile) index files. In some cases this is not sufficient. See the note on APT below for a workaround.

2. Store copies of your `.debs`, `.orig.tar.gz`, ... somewhere in the `_import` subdirectory in the cache, ie. in `/var/cache/apt-cacher/_import/`. The files may be links or symlinks, does not matter. When done, apt-cacher will move those files to its own internal locations. Example:

```
cd /var/cache
mkdir apt-cacher-ng/_import
cp -laf apt-proxy apt-cacher /var/cache/apt-cacher-ng/_import
chown -R apt-cacher-ng apt-cacher-ng/_import
```

3. Visit the report page and trigger the import action there. Check the results, look for (red) error messages.
4. Check the `_import` directory again. All files that could be identified as referenced by archive metadata should no longer be there if they have been successfully moved. If some files have been left behind, check whether the client can use them, i.e. with "apt-cache policy ..." and/or checking checksums with `md5sum/sha1sum` tools. Probably they are no longer needed by anyone and therefore apt-cacher-ng just left them behind. If no, follow the instructions in 1 or do similar things for your distribution and retry the import operation. Setting the verbosity flag (see checkbox on the command-and-control page) can also help to discover the reason for the refusal to import the particular files.

NOTE: APT is pretty efficient on avoiding unnecessary downloads which can make a proxy blind to some relevant files. ACNG makes some attempts to guess the remote locations of missed (not downloaded) files but these heuristics may fail, especially on non-Debian systems. When some files are permanently ignored, check the process output for messages about the update of Packages/Sources files. When some relevant package sources are missing there, there is a brute-force method for Debian/Ubuntu users to force their download to the client side. To do that, run:

```
rm /var/cache/apt/*cache.bin
rm /var/lib/apt/lists/*Packages
rm /var/lib/apt/lists/*Sources
```

on the client to purge APT's internal cache, and then rerun "apt-get update" there.

## 8.2 Cache overview

To get a basic overview of the cache contents, the distkill.pl script may be used. See section 7.2 for details and warnings.

```
# /usr/lib/apt-cacher-ng/distkill.pl
Scanning /var/cache/apt-cacher-ng, please wait...
Found distributions:
1. testing (6 index files)
2. sid (63 index files)
3. etch-unikl (30 index files)
4. etch (30 index files)
5. experimental (505 index files)
6. lenny (57 index files)
7. unstable (918 index files)
8. stable (10 index files)
```

WARNING: The removal action would wipe out whole directories containing index files. Select d to see detailed list.

Which distribution to remove? (Number, 0 to exit, d for details): d

Directories to remove:

1. testing:  
  /var/cache/apt-cacher-ng/debrep/dists/testing
2. sid:  
  /var/cache/apt-cacher-ng/localstuff/dists/sid  
  /var/cache/apt-cacher-ng/debrep/dists/sid
4. etch:  
  /var/cache/apt-cacher-ng/ftp.debian-unofficial.org/debian/dists/etch
5. experimental:  
  /var/cache/apt-cacher-ng/debrep/dists/experimental
6. lenny:  
  /var/cache/apt-cacher-ng/security.debian.org/dists/lenny  
  /var/cache/apt-cacher-ng/debrep/dists/lenny
7. unstable:  
  /var/cache/apt-cacher-ng/debrep/dists/unstable  
  /var/cache/apt-cacher-ng/localstuff/debian/dists/unstable
8. stable:

```
/var/cache/apt-cacher-ng/debrep/dists/stable
Found distributions:
```

WARNING: The removal action would wipe out whole directories containing index files. Select d to see detailed list.

## 8.3 Access to SSL/TLS remotes (HTTPS)

It is possible to have encrypted access to remote sites via HTTPS protocol with recent versions of apt-cacher-ng if the OpenSSL support was enabled at compile time. However this leads certain side effects and complications; due to the nature of the HTTPS connection model, it is not possible to act as an intermediate server (e.g. caching proxy) by the same rules as with HTTP:

- SSL ensures strict verification of remote host name. Therefore, you cannot add an URL to sources.list and make them point to "hostname of acng server": the download client would expect the remote site to offer the certificate of "hostname of acng server" but the certificate of the remote server would be used.
- This restriction also applies if the LAN admin somehow manages to trick the client that it connects to the remote site but the real connection is silently rerouted to the local server: since the proxy in the middle cannot fake the real certificate it would need to create a custom one but that manipulation usually cannot go unnoticed by the client unless some local hack disables or manipulates the certificate verification. All parts of this should be considered a crude hack and won't be described here in detail. In fact, apt-cacher-ng does not implement local SSL server functionality (as of version 0.8.1), partly because of these considerations.
- If the client assumes that it needs to access a HTTPS remote server through a HTTP proxy, it usually goes for the HTTP tunneling (see the Wikipedia article for details). This is a well-known and widely-used method but it comes with some disadvantages: the proxy has no control of the data flowing through it so the proxy cannot cache the data in any useful way; in environments with additional security restrictions, it's also hard to identify malicious users that abuse the service for illegitimate purposes.

Considering these difficulties, there are three (and a half) methods to use SSL.

- First, the "half method" - not using the proxy at all, configuring each client to not use the HTTP proxy for HTTPS urls. This will obviously disable central caching and requires the client has separated configuration options to set this. For Debian based distros, this can be done by adding a configuration like this: `Acquire::https::proxy "DIRECT" ;` to apt.conf or one of the apt.conf.d files. See section 8.5 for further information.
- The "backend configuration method": if the clients access the remote sites through URLs remapped on the server, the cacher admin can add https URLs to backend lists instead of http urls. Data will be cached just like usual.
- The "laissez-faire method": in acng.conf (or related) configure the `PassThroughPattern` option to contain a regex like `.*` and configure the clients to use apt-cacher-ng as HTTP proxy and let the clients connect to https URLs "as usual". Some limited access control can be achieved through adjustment of the regular expression (`.*` permits access to any host and any port, including 443 for https). Data is not cached on the server.



- The "tell-me-what-you-need method": on the client side, modify the access URLs and change https to http and put the string "HTTPS///" between http:// and the host name. With this trick, the user client will access the proxy like going for a usual HTTP download and the proxy will access the remote URL with the https protocol. Caching (and file merging to repositories) will work and there is still enough flexibility for the users. The disadvantages of this method are basically the same as with the access URLs rewriting (see Section 3.2) but the method is still the preferred one by the apt-cacher-ng author. For apt's sources.list, the modification may look like the following example.

```
deb http://HTTPS///get.docker.com/ubuntu docker main
# If apt-cacher-ng is configured as proxy in APT, this makes it
# switch internally to https://get.docker.com/ubuntu
deb http://acnghost:3142/HTTPS///get.docker.com/ubuntu docker main
# Basically the same just with access to apt-cacher-ng through
# URL rewritting instead of setting http proxy.
```

## 8.4 JIGDO usage

It's possible to use apt-cacher-ng source with the jigdo-lite utility. There are some limitations, though:

- since many mirrors do not distribute the jigdo files (or even nothing from cdimage.debian.org at all), there is a high chance to be redirected to a such mirror when using the backend-mapped configuration. I.e. when user follows the official documentation and edits wgetOpts in the jigdo configuration, it will fail in many cases.
- apt-cacher-ng does not support .template files properly. They might be cached but will be expired (removed from cache), sooner or later.

But it's possible to feed jigdo-lite with the package contents from your mirror. To do that, first start jigdo-lite as usual, something like:

```
jigdo-lite http://cdimage.debian.org/.../...-DVD-1.jigdo
```

When asked about Debian mirror, enter something like:

```
http://proxy.host:3142/ftp.de.debian.org/debian/
```

i.e. construct the same URL as present in usual apt-cacher-ng's user's sources.list.

That's all, jigdo-lite will fetch the package files using apt-cacher-ng proxy.

## 8.5 Avoid use of apt-cacher-ng for certain hosts

Sometimes clients might need to access some remote side directly to do some non-file-transfer oriented work but still passing the data through configured apt-cacher-ng proxy. Such remote hosts can be marked for direct access in apt configuration, e.g. in /etc/apt/apt.conf:

```
Acquire::HTTP::Proxy::archive.example.org "DIRECT";
//or Acquire::HTTP::Proxy::archive.example.org "other.proxy:port"
```

## 8.6 Avoid caching for certain domains or certain file types

Sometimes clients to download through apt-cacher-ng but the data shall not be stored on the harddisk of the server. To get it, use the DontCache directive (see examples for details) to define

such files.

## 8.7 How to make big download series faster

Symptom: A common situation is a periodic download of hundreds of files through apt-cacher-ng where just a half is present in the cache. Although caching works fine, there are visible delays on some files during the download.

Possible cause and relief: the download from the real mirror gets interrupted while apt-cacher-ng delivers a set of files from the internal cache. While the connection is suspended, it times out and needs to be recreated when a miss occurs, i.e. apt-cacher-ng has to fetch more from the remote mirror. A workaround to this behaviour is simple, provided that the remote mirror can handle long request queues: set the pipelining depth to a very high value in apt.conf file or one of its replacement files in /etc/apt/apt.conf.d/. With something like:

```
Acquire::http { Pipeline-Depth "200"; }
```

there is a higher chance to get the server connection "preheated" before a stall occurs.

## 8.8 How to import DVDs or ISO images

First, it should be clear what is needed to be done. In order to integrate the packages from a DVD or ISO image, read on in section 8.9.

The situation with ISO files import is complicated. They are not supported by the cache and there is also no expiration mode for them. The feature might be considered for addition in some future release of apt-cacher-ng.

What is possible now is publishing a directory with ISO files using its web server mode, see LocalDirs config option for details.

## 8.9 How to integrate DVDs or ISO image data

Integrating package files from DVD or ISO images is not much different to the usual import operation, see above for instructions.

One possible way to get files into the `_import` directory is simply mounting it there:

```
mount -o loop /dev/cdrom /var/cache/apt-cacher-ng/_import
```

After running the import operation, the disk can be umounted and removed.

A possible variation is import via symlinks. This can make sense where the space consumption must be reduced and the ISO image should stay on the server for a long time. To achieve this, the image should be mounted at some mount point outside of the `_import` directory; the mounted state should be made permanent, maybe via an `/etc/fstab` entry with the `loop` option; then a symbolic link tree pointing to the mountpoint location should be created in the `_import` directory (something like `cp -as /mnt/image_jessie_01/pool /var/cache/apt-cacher-ng/_import/`). The subsequent "import" operation should pick up the symlinks and continue using them as links instead of file copies.

## 8.10 How to execute commands before and after going online?

It is possible to configure custom commands which are executed before the internet connection

attempt and after a certain period after closing the connection. The commands are bound to a remapping configuration and the config file is named after the name of that remapping config, like `debrep.hooks` for `Remap-debrep`. See section 4.3.2, `conf/* .hooks` and `/usr/share/doc/apt-cacher-ng/examples/* .hooks` files for details.

## 8.11 Listen to only specific interfaces or IP protocols

Unless configured explicitly, the server listens to any interface with IPv4 or IPv6 protocol. To disable some of this, use the `BindAddress` option. It should contain a list of IP addresses associated with particular network interfaces, separated by space. When option is set then the server won't listen to addresses or protocols not included there.

To limit to specific IP protocol, the address should only be present in the protocol specific syntax (like `192.0.43.10`) will limit the use to the specific protocol.

The usual wildcard addresses can also be used to match all interfaces configured for the specific protocol, like `0.0.0.0` for IPv4.

## 8.12 How to avoid use of IPv4 (or IPv6) where possible?

Usually, outgoing hosts are accessed by the protocol and with the target IP reported as the first candidate by operating system facilities (`getaddrinfo`). It is possible to change this behavior, i.e. to skip IPv6 or IPv4 versions or try IPv6 connection first and then use IPv4 as alternative (or vice versa). See option *ConnectProto* in configuration examples.

## 8.13 Use the proxy without storing all data twice

There is a general use case where the data storing behavior of APT is not so fortunate. Imagine an old laptop with a slow and small harddisk but a modern network connection (i.e. Cardbus-attached WLAN card). But there is not enough space for APT to store the downloaded packages on the local disk, or not enough to perform the upgrade afterwards.

A plausible workaround in this case are moving contents of `/var/cache/apt/archives` directory to a mounted NFS share and replacing the original directory with a symlink (or bind-mount to the mentioned share). However, this solution would transfer all data at least three times over network. Another plausible workaround might be the use of `curlftpfs` which would embed a remote FTP share which then can be specified as `file:// URL` in `sources.list`. However, this solution won't work with a local HTTP proxy like `apt-cacher-ng` (and `httpfs` `http://sourceforge.net/projects/httpfs/` is not an alternative because it works only with a single file per mount).

As real alternative, `apt-cacher-ng` comes with an own implementation of a http file system called `acngfs`. It makes some assumptions of proxy's behaviour in order to emulate a real directory structure. Directories can be entered but not browsed (i.e. content listing is disallowed because of HTTP protocol limitations). Anyhow, this solution is good enough for APT. When it's checking the contents of the data source located on `acngfs` share, it reads the file contents of just the files required for the update which makes the `apt-cacher-ng` server download them on-the-fly.

And finally, `angfs` usage can be optimized for local access. This works best if the proxy daemons runs on the same machine as `acngfs` and there are hundreds of packages to update while filesystem access costs are negligible. Here the cache directory can be specified in `acngfs` parameters, and then it gets files directly from the cache if they are completely downloaded and don't have volatile contents.

## 8.14 Partial Mirroring

It is possible to create a partial local mirror of a remote package repository. The method to do this is usually known as pre-caching. A such mirror would contain all files available to apt through `apt-cacher-ng`, making the cache server suitable for pure off-line use.

The config uses index files in the local cache in order to declare which remote files shall be mirrored. Choice of relevant files decides which branch, which architecture or which source tree is to be mirrored. For convenience, it's possible to use glob expressions to create semi-dynamic list. The format is shell-like and relative to cache directory, a shell running in the cache directory can be helpful to verify the correctness.

*Example:*

```
PrecacheFor: debrep/dists/unstable/*/binary-amd64/Packages*
```

```
PrecacheFor: emacs.naquadah.org/unstable/*
```

Assuming that debrep repository is configured with proper remapping setup (see above), this would download all Debian packages listed for amd64 architecture in the unstable branch.

There is also support for faster file update using deltas, see `Debdelta` for details. The `delta_uri` URL mentioned there needs to be added as `deltasrc` option, see section 4.3.2 for details.

The operation is triggered using the web interface, various options or estimation mode can also be configured there. The CGI URL generated by the browser can be called with other clients to repeat this job, for example in a daily executed script. Another possible command line client can be the `expire-caller.pl` script shipped with this package (replacing the CGI parameters through environment, see section 7.1.2). For regular tools like `wget` or `curl`, remember the need of quotation and secrecy of user/password data - command calls might expose them to local users.

# Chapter 9: Troubleshooting

---

## 9.1 Debugging

Preliminary meanings of Debug option settings are:

- 0: No debug printing
- 1: Log file buffers are flushed faster
- 2: Some additional information appears within usual transfer/error logs
- 4: extra debug information is written to apt-cacher.err (also enables lots of additional trace points when apt-cacher-ng binary is built with debug configuration, see section 9.6 for details)

To combine that settings, add them (i.e. 7 enables all messages and log flushing)

Getting HTTP headers from apt-get works like this:

```
apt-get update -o Debug::Acquire::Http=true
```

## 9.2 Problem: keeps delivering damaged files

Even in this millennium, sometimes damaged files are downloaded from the server and are stored in the cache. Sometimes lazy maintainers of 3rd party archives replace package files with the same name but different contents. Sometimes the server's file system gets corrupted without detection by the OS.

Anyhow, there might be cases where cached data becomes invalid. Volatile files might be replaced by fixed version on some future download but static package files are never changed upon completion and even incomplete downloads are resumed and keep bad data downloaded before.

Usually the damage is only discovered by the client later. The particular file can be located in the cache and replaced manually. And if there are many of them, a mass file check might be needed to clean the mess. Fortunately, there are helpers in cache maintenance interface to automate this process.

To start, visit the web control interface and check the options of *Expiration* task. Enable the check for explicit paths and the check of data contents, then start the expiration. With this parameters, complete files with incorrect checksum are detected. The default action for such files is adding them to a list of damage files. After that, the "Delete damaged files" button in the main web page can be used to remove them (or the Show button to display them first). Alternatively, the checkboxes appearing aside of each damage detection can be used together with the control buttons which appear at the end of the report. And another way of dealing with them is truncating

(setting to zero size). This can be done on-the-fly and is enabled by the expiration parameters, or with the appropriate command button in the web interface.

NOTE: several index files and related support files can create false positives, i.e. as incomplete or bad files. This usually happens because their volatile contents has changed but the file was not downloaded for a while and another version of it was used instead (like bzip2-compressed instead of gzip-compressed or uncompressed). The default code attempts to detect files with good reasons to stay in the cache and does not mark them as damaged.

### 9.3 Problem: regular expiration action reproducibly aborts

A quick investigation of action logs should help identifying the problem. A typical one is a mirror listed somewhere which is not reachable when expiration runs.

Unfortunately there is no simple and safe way to solve this. One method is setting the `ExAbortOnProblems` configuration variable, but this can destroy the whole cache if a bigger problem with index file occurs and this state remains unnoticed for many days until `ExTreshold` period (see configuration) is over.

Another way is listing the index files of the faulty mirrors to a special file. It needs to be stored as "ignore\_list" in the configuration directory and store one path name per line with paths relative to the cache directory, as seen in the error messages.

### 9.4 Problem: cacher suddenly terminates, log reports IO errors

For simplicity and memory saving reasons, apt-cacher-ng assumes that some files can be opened exclusively for reading and they don't suddenly become unreadable. Unfortunately, in some conditions and in case of IO errors, this doesn't work and the file systems sends a fatal signal and eventually terminates the program.

To track this down to the likely reasons, it's possible to execute a custom script in the moment of the reception of the fatal signal. To do that, add something like this to the configuration:

```
BusAction = ls -l /proc/$PPID/ | mail -s SIGBUS! root
```

This command would send a list of opened files with paths to the "root" mail user, and the root shall check the state of those files (maybe run them through md5sum or similar).

NOTE: Beware of the security implications of this configuration option. It runs regular shell code in context of the daemon user in a blocking way, so it may be vulnerable to symlink attacks and it will delay the automatic restart of the daemon through systemd.

### 9.5 Problem: download fails with 503 ... status message

Code 503 usually represents an internal failure which could not be described correctly by other HTTP status codes. In the most cases it's caused by file system errors or incorrect cache directory setup, like files or directories with incorrect owner, missing write/read permissions for the effective user account or other system related exceptions like running out of disk space.

The log file `apt-cacher.err` located in the `LogDir` directory should document more details. In case it doesn't, setting the `Debug` config option to a higher value might reveal more information.

Fixing permission problems shouldn't be a real challenge for system administrators. Usually, a

command set like this should do the trick on Debian/Ubuntu systems, assuming that all group users should receive write access to the cache files:

```
chown -R apt-cacher-ng:apt-cacher-ng /var/cache/apt-cacher-ng
chmod -R a+rX,g+rw,u+rw /var/cache/apt-cacher-ng
```

## 9.6 Problem: apt-get freezes when downloading files

Solution: First, check:

- Free disk space and inode usage (`df`, `df -i`)
- Internet connection to the remote sites (browse them via HTTP, e.g. visiting `http://ftp.your.mirror`)

If nothing helps then you may have hit a spooky problem which is hard to track down. If you like, help the author on problem identification. To do that, do:

```
su -
# enter root password
cd /tmp
apt-get source apt-cacher-ng
apt-get build-dep apt-cacher-ng
cd apt-cacher-ng-*
make distclean all DEBUG=1
/etc/init.d/apt-cacher-ng stop
./apt-cacher-ng -c /etc/apt-cacher-ng logdir=/tmp foreground=1 debug=7
# (let apt-get run now, on timeouts just wait >> 20 seconds)
# stop the daemon with Ctrl-C
/etc/init.d/apt-cacher-ng start
# compress /tmp/apt-cacher.err and send it to author
chown -R apt-cacher-ng:apt-cacher-ng /var/cache/apt-cacher-ng
```

The value of debug can be varied to have different verbosity (see section 9.1 for more information about Debug levels).

## 9.7 apt-get reports corrupted bzip2 data

Symptoms: apt-get fails to run through "update" no matter what you do. And you may have get a message like this one.

```
99% [6 Packages bzip2 0] [Waiting for headers] [Waiting for headers]
bzip2: Data integrity error when decompressing.
      Input file = (stdin), output file = (stdout)
```

It is possible that the compressed file(s) have become corrupted. You can use the `-tvv` option to test integrity of such files.

You can use the `'bzip2recover'` program to attempt to recover data from undamaged sections of corrupted files.

```
Err http://debian.netcologne.de unstable/main Packages
Sub-process bzip2 returned an error code (2)
```

- This might be one of Apt's problem with insufficient handling of errors, i.e. passing incomplete files to bzip2 on premature connection termination. Retry the update and it might work.
- Another issue is more severe: old versions of apt-cacher-ng had a bug which could cause data corruption while resuming downloads however this problem appears only in unusual conditions. To make sure there are no broken files in your repository, run the Expiration task with content verification enabled, and also "immediate deletion" (or delete later after checking the list). See section 7.1.1 for details.

## 9.8 Problem: apt-cacher-ng refuses to start with "Address already in use"

Another service is already listening on the port which apt-cacher-ng is configured to use. This might be the apt-cacher daemon which used the same port number by default. To identify the daemon behind that process, use the fuser utility, executing it as root for IPv4 and IPv6 protocol versions. Example:

```
fuser -4 -v -n tcp 3142
fuser -6 -v -n tcp 3142
          USER          PID ACCESS COMMAND
3142/tcp:      xwwwfsd    17914 F....  xwwwfsd
```

(where 3142 is the port number from the apt-cacher-ng configuration file). To resolve the collision, reconfigure the other daemon or apt-cacher-ng to use another free port (and reconfigure the clients to use the new apt-cacher-ng port accordingly).



## Chapter 10: Known Bugs and Limitations

---

- Only the "Basic" type of HTTP proxy authentication is supported at the moment
- Only HTTP HEAD and GET commands are supported properly.
- CONNECT (for https/SSL/TLS tunnels) is supported but needs to be enabled by local administrators, see section 8.3 for details
- POST support is restricted to apt-listbugs actions and is actually implemented as simple data forwarding with minor header manipulation. If additional proxy is configured, the client will be redirected to the HTTPS version of bugs.debian.org service, i.e. switching to the CONNECT method (see above)
- Transparent proxy mode not implemented yet
- The metadata update in maintenance jobs (like expiration task) attempts to guess the best source of update based on original source information and current backend configuration. In case of repeating failures, please run a "forced" download of index files through the web interface
- See TODO file in apt-cacher-ng source for more related information

## Chapter 11: Contact

---

The planned features are listed in the file TODO. Don't hesitate to contact the author if you really need something not found there and you can explain the severity of your request.

There are also a public mailing list and request trackers available on the Alioth project page.

And last, but not least: feel free to express your gratitude by donating a small amount of money via PayPal (to [edi@gmx.de](mailto:edi@gmx.de)) or just Flattr it!