

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	4
Chapter 2: Running apt-cacher-ng	5
Chapter 3: Basic Configuration	6
3.1 Server Configuration	6
3.2 Client Configuration	6
Chapter 4: Advanced Server Configuration	7
4.1 Vocabulary	7
4.2 Configuration file types	7
4.3 Repositories and URL mapping	8
4.3.1 Basic use of URL remapping	8
4.3.2 Details of remapping syntax	9
Chapter 5: Security	11
Chapter 6: Maintenance	12
6.1 Manual cache cleanup	12
6.2 Automated cache cleanup	13
6.3 Distribution release removal	14
Chapter 7: HOWTOs and FAQ	15
7.1 Package import	15
7.2 Cache overview	16
7.3 Access control and inetd usage	17
7.4 JIGDO usage	17
7.5 Debugging	18
Chapter 8: Troubleshooting	19
8.1 Problem: <i>apt-listbugs</i> reports errors using HTTP proxy	19
8.2 Problem: non-interactive expiration action reproducibly aborts	19

8.3 Problem: <i>apt-get</i> freezes when downloading files	19
8.4 <i>apt-get</i> reports corrupted bzip2 data	20
8.5 Problem: APT client receives a "Cache storage error"	20
8.6 Problem: <i>apt-cacher-ng</i> refuses to start with "Address already in use"	20
Chapter 9: Known Bugs and Limitations	22
Chapter 10: Contact	23

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 package 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, other aspects have been declared as primary targets during the development of apt-cacher-ng:

- lightweight implementation - allow the 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 excessive use of hidden file system features for internal operations
- real (effective) support of HTTP pipelining, therefore a native client with native stream control has been developed. The nice side effect is the reduction of resource overhead and minimization of possible points of failure
- avoiding featurities 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 should be 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-parseable 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`

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.`

Chapter 3: Basic Configuration

3.1 Server Configuration

Unlike some rumors on the internet tell people, 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 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.

3.2 Client Configuration

From the client side, apt-cacher-ng can be used as 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 APT, e.g. 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
```

now would become:

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

(assuming that CacheServerIp is 192.168.0.17).

Mixing both configuration methods is not recommended and will lead to obscure APT failures in most cases.

Additionally, leading path component containing "apt-cacher/" or "apt-cacher?/" might be ignored by the server during the URL processing. This is intended behavior and exists to maintain backwards compatibility to `sources.list` entries configured for early versions of Apt-Cacher (based on CGI technology).

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.

- "Repository": the internal identifier of a local cache directory. Can be the hostname of an actual mirror or an arbitrary name for a ring of mirror servers, in which case you need to provide a backend definition (see below). When chosen once, the repository string should not be changed afterwards or the data in the cache might become inaccessible to the client.
- "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 repository is classified as 'volatile'. This includes Packages, Sources, Release, Pdiff and similar files. Program messages sometimes refer to them as 'index files'.
- "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 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.

The main configuration files are located in one single directory. Three types files are recognized (by suffix) and interpreted by the following schema.

*.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).

Lists of remote package repositories (i.e. mirrors) can be specified in one of two 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/"). An URL must start with http:// and should end with a slash.
- in 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.

Apt-cacher-ng autodetects the format of the list.

4.3 Repositories and URL mapping

IMPORTANT: it should be clear from the beginning that the URL remapping is an optional feature. You do not have to use it, but you can. It is not a strong prerequisite for apt-cacher-ng's work and in trouble, this feature can be disabled for debugging purposes. On the other hand, it's also possible to forbid users to download from non-remapped locations using the `ForceManaged` option.

4.3.1 Basic use of URL remapping

URL remapping has several uses. First, it allows your apt-cacher-ng based proxy to masquerade as a Debian or Ubuntu mirror; it can be made to appear to contain the same directory structure and files the client would find on a real mirror. Instead of referencing a real mirror, the client's `sources.list` file would contain a line like `deb http://name.of.proxy.host/debian stable main`. Your backend definition in apt-cacher-ng's configuration determines which actual Debian mirror will be used to fetch the files.

This keeps the client configuration clean, simple and straightforward. Without URL remapping, the `sources.list` file on the client would have to read something like `deb http://name.of.proxy.host/ftp.de.debian.org/debian stable main` or alternatively additional setting of HTTP proxy for APT would be required. Note that, depending on the remapping configuration, it is possible that apt-cacher-ng will actually use a mirror other than the one requested by the client (see below).

A second use is obscuring the real location of a repository from your clients. This allows you to change the real location easily, without having to modify the configuration of all clients; you just have to edit the backend configuration in one single place.

What happens behind the scenes can be demonstrated in few examples. In all examples below, fetching from `http://proxy.host/some-url` is the same as fetching from `http://some-url` having APT-Configuration with proxy-host as HTTP proxy. The exact meaning of the configuration directive(s) is explained below.

Example 1, no remapping is used:

When a client asks for `http://proxy.host/ftp.de.debian.org/debian/something`, apt-cacher-ng fetches `http://ftp.de.debian.org/debian/something`.

Example 2:

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


This is a simple case. When a client asks for `http://proxy.host/ftp.de.debian.org/debian/something`, `apt-cacher-ng` fetches `http://ftp.de.debian.org/debian/something`. When another client asks for `http://proxy.host/ftp.at.debian.org/debian/something` (i.e. the same "something" from another mirror) then the file is delivered from the local cache (if cached, otherwise it would be fetched from `ftp.at.debian.org`). Which means, `apt-cacher-ng` uses the file which was originally downloaded from another mirror; they are considered equal as advised by the `Remap-...` directive above.

Example(s) 3, slightly more complex:

```
Remap-ubuntu: /ubuntu ; http://us.archive.ubuntu.com/ubuntu
Remap-medibuntu: /medibuntu ; http://packages.medibuntu.org
```

These two examples specify trivial mappings. Whenever a client asks for `http://proxy.host/ubuntu/foo`, `apt-cacher-ng` will retrieve `http://us.archive.ubuntu.com/ubuntu/foo`. When a client wants `http://proxy.host/medibuntu/bar`, `apt-cacher-ng` fetches `http://packages.medibuntu.org/bar`. This is still relatively simple.

This game can be continued: adding a path before semicolon adds more URL variants "visible" by the client (see Example 2), and it's also possible to add multiple backend URLs (after semicolon). When multiple backends are set, they are probed in the specified order until the download succeeds.

Side note: "success" means a started download or a non-critical failure in this context, i.e. a "404 File not found" status is not a critical failure, since APT (client) sometimes looks remote file's existence and reacts to such status response accordingly.

4.3.2 Details of remapping syntax

A "Remap-" line contains three pieces of information:

1. A "repository name": unique internal identifier for the remapping ruleset specified by the line; in the examples above, "ubuntu" and "medibuntu". These identifiers have no meaning; we could just as well have chosen "blarg" and "deedledum". Restrictions: the identifier must be valid as a filename; it mustn't begin with an underscore; and it mustn't contain whitespace. It should also not be "apt-cacher" for historical reasons.

This is the part of the line between "Remap-" and the first colon (':').

2. A list of path (first URL parts) to apply remapping to. This is the part of the line between the first colon and the first semicolon(';'). The simple way is specifying the URL paths directly, separated by spaces. To keep the configuration line short with more than few mirrors, they can be listed in a separate file and this list file can be specified in the Remap line instead of the URL path list. I.e. you can use "file:filename" to read a list of pathnames from "filename". Filename can be absolute, or relative to the configuration directory. You can mix files and literal pathnames in the list. Pathnames mustn't contain wildcards, but "file:" specifications may. `apt-cacher-ng` can decompress .gz and .bz2 files. A leading "http://" in pathnames is replaced by a single "/".

3. A list of backend servers to fetch the files from, whenever clients request a file from a remapped path. Backend lists are specified the same way as path lists.

Complex example:

Remap-

```
debrep: file:deb_mirror*.gz /debian ; file:backends_debian
```

This causes apt-cacher-ng to read a list of debian mirrors from all files matching "deb_mirror*.gz" and construct a list of pathnames consisting of entries like /ftp.de.debian.org/debian /ftp.kfki.hu/linux/debian and so on. The last element of the list will be simply /debian. All these paths will be remapped to paths read from backends_debian, which can contain the base URL of a single Debian mirror, or several URLs, one per line. The same RFC822-like format deb_mirrors.gz is supplied in is also supported, so you could just make backends_debian a subset of deb_mirrors.gz; however the use of good local mirrors is recommended.

Client sources.list files could specify either of the following and still actually use the mirrors from backends_debian:

- deb http://proxy.host/arbitrary.official.debian.mirror/basepath stable main
- deb http://proxy.host/debian stable main
- deb http://arbitrary.official.debian.mirror/basepath stable main (with APT configuration for HTTP proxy, see section 3.2).

Side notes:

- Multiple "Remap-samename" directives are possible (sharing the same repository key). In this case their contents are merged.
- If no repository is specified, the local storage space is derived from the request URL. If no backend is specified then the internal download uses the site in the request URL to download from. If one or multiple backends are specified then the remote site and real location are calculated from the backend description.
- Use of predefined repositories is recommended. Large remapping lists (for many popular Debian mirrors) are supported quite efficiently; assigning them all into the same repository increases the probability of sharing the data between users who where going to use different mirrors. The backends list can be customized to ensure that few locally best reachable mirrors are used to download from.

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 executed on.

As to the program internal security, apt-cacher-ng has been developed to care about a certain level of attacks from internal users as well as from malicious outside hosts. However, no guarantees can be made about the security of the program. It's recommended to run apt-cacher-ng under a system account which has no access to any system files outside of the cache and log directories. Refer to the manuals of the administration utilities of your 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 the `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).

Chapter 6: Maintenance

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

6.1 Manual cache cleanup

If a package is no longer downloadable by APT clients then its files are also not referenced in any volatile (index) file and can be removed. This rule also applies to most volatile files at the distribution level. I.e. the 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, ...).

To run this 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*:

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 contents where possible, also attempt to detect incorrect file size information in the cached metadata. Note: the check results are stored only once, future calls without this option can overwrite the results again. Use action buttons (see below) to delete corrupted files after the scan.

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 below).

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!

6.2 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, also see the cron script for details:

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.

6.3 Distribution release removal

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 assumed index file trees, providing the option to remove some immediately. The script should be used with extreme care! See section 7.2 for example of its output.

Chapter 7: HOWTOs and FAQ

7.1 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 ("apt-cache show" and checking checksums with `md5sum/sha1sum` tools). If yes, then repeat the procedure and be more careful in step 1. If that doesn't work, enable verbose output in the web interface and watch out for error messages.

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 to force their download to the client (for clients with Debian only). 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.

7.2 Cache overview

To get a basic overview of the cache contents, the distkill.pl script may be used. See section 6.3 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.

7.3 Access control and inetd usage

Filtering by client IP or hostname is not supported directly. However, 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).

7.4 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.

7.5 Debugging

Preliminary meanings of Debug option settings are:

- 0: No debug printing
- 1: Log file write buffers are flushed faster
- 2: extra information appears within usual transfer log
- 3 and 4: limited debug information is written to apt-cacher.err
- 5: much more debug information written to apt-cacher.err (requires a special binary, see above)
- 11: like 5 but all Debug marks are printed to the log, including anonymous marks for code flow tracking.

Getting HTTP headers from apt-get works like this: `apt-get update -o Debug::Acquire::Http=true`

Chapter 8: Troubleshooting

8.1 Problem: *apt-listbugs* reports errors using HTTP proxy

Solution: advise them not to use the main proxy for access to bugs.debian.org, adding some bits like following to */etc/apt/apt.conf*:

```
Acquire::HTTP::Proxy::bugs.debian.org "DIRECT";  
//or Acquire::HTTP::Proxy::bugs.debian.org "other.proxy:port"
```

Real configurable pass-through for unsupported actions mode will be added in some future version of Apt-Cacher NG.

Thanks to Jamil Djadala for finding this workaround.

8.2 Problem: non-interactive 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.

8.3 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 acng DEBUG=1
/etc/init.d/apt-cacher-ng stop
./apt-cacher-ng -c /etc/apt-cacher-ng logdir=/tmp foreground=1 debug=6
# (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 7.5 for more information about Debug levels). 11 is recommended for really weird problems.

8.4 *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 6.1 for details.

8.5 Problem: APT client receives a "Cache storage error"

Examine the last entries in the apt-cacher.err file in the log directory. Most likely it's caused by wrong permissions of some directory in the cache folder where new files/directories need to be created, or by insufficient disk space.

8.6 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).

Chapter 9: Known Bugs and Limitations

- Versions between 0.2.6 and 0.3.3 created broken X-Original-Source URLs in the .head files
- Only HTTP POST and GET command are supported (no POST, no PUT), this make SOAP calls through apt-cacher-ng impossible
- HTTP redirection is not supported yet
- Transparent proxy support not implemented yet
- diff-based reconstruction of index files is not implemented
- See TODO file in apt-cacher-ng source for various other notes
- "Code 520824" (see below)

Few versions of apt-cacher-ng did loose some information about the original source of downloaded files. This has been unnoticed for many months, because the path resolution algorithm tries to use a file path components go guess the original URL, or guess the cache repository and then download from using this repository (backends definitions).

The good news is that the missing information is completed automatically when a file is redownloaded once (or resumed) through apt-cacher-ng. Static package files are usually not redownloaded, but the missing information is not relevant for them.

The bad news is that in some cases this might become a problem, particularly when an expiration/import tasks tries to refresh the index file which is stored inside of a (former) repository tree and the administrator has removed the backends definitions for this repository. When ACNG runs out of possible sources it reports "Code 520824" and refers to this text. However, the detection of this situation is still not perfect and in rare cases it's possible that messages about failed resolution of some hostname (identical with former cache repository name) will appear.

Chapter 10: Contact

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

Please report any bugs found in apt-cacher (which are not obviously the complementary of missing features, see above ;-).