apt-smart

Release 7.1.3

May 31, 2020

Contents

Welcome to the documentation of *apt-smart* version 7.1.3!

Source code: https://github.com/martin68/apt-smart

The following sections are available:

CHAPTER 1

User documentation

The readme is the best place to start reading, it's targeted at all users and documents the command line interface:

1.1 apt-smart: Smart, automated Debian/Ubuntu/Linux Mint mirror selection

The *apt-smart* package automates robust apt-get mirror (a.k.a Repositories, Sources) selection for Debian , Ubuntu and Linux Mint by enabling smart discovery of available mirrors, smart ranking of available mirrors, automatic switching between mirrors and robust package list updating (see *features*). It's currently tested on Python 2.7, 3.4, 3.5, 3.6, 3.7, 3.8 and PyPy (although test coverage is still rather low, see *status*).

1.1.1 Why?

As a successor of apt-mirror-updater, *apt-smart* has many improvements in intelligence, speed, accuracy and robustness (see changelog) when offering the best mirror for you. It has a plan to optionally be a set-and-forget smart daemon: running in the background as a reverse proxy always redirecting to the best mirror without root privilege. It also has a plan to support other distros like: Linux Mint (Done!) , ROS. . .

1.1.2 Features

Smart discovery of available mirrors Debian , Ubuntu and Linux Mint mirrors are discovered automatically by querying the Debian mirror list or the Ubuntu mirror list1 or the Ubuntu mirror list2 or the Linux Mint mirror list (the applicable mirror list is automatically selected based on the current platform). It can smartly get mirrors within the country which the user is in.

Smart ranking of available mirrors Discovered mirrors are ranked by bandwidth (to pick the fastest mirror) and whether they're up-to-date and excluded if they're being updated (see *issues with mirror updates*). e.g. with *–list-mirrors* flag it would output like this:

```
-------------------------------------------------------------------------------------------
↪---------------
| Rank | Mirror URL                       | Available? | Updating? | Last updated    ␣
↪| Bandwidth     |
-------------------------------------------------------------------------------------------
↪---------------
|    1 | http://archive.ubuntu.com/ubuntu | Yes        | No        | Up to date      ␣
↪| 16.95 KB/s    |
|    2 | http://mirrors.cqu.edu.cn/ubuntu | Yes        | No        | 3 hours behind  ␣
↪| 427.43 KB/s   |
|    3 | http://mirrors.nju.edu.cn/ubuntu | Yes        | No        | 5 hours behind  ␣
↪| 643.27 KB/s   |
|    4 | http://mirrors.tuna.tsinghua.e...| Yes        | No        | 5 hours behind  ␣
↪| 440.09 KB/s   |
|    5 | http://mirrors.cn99.com/ubuntu   | Yes        | No        | 13 hours behind␣
↪| 2.64 MB/s     |
|    6 | http://mirrors.huaweicloud.com...| Yes        | No        | 13 hours behind␣
↪| 532.01 KB/s   |
|    7 | http://mirrors.dgut.edu.cn/ubuntu| Yes        | No        | 13 hours behind␣
↪| 328.25 KB/s   |
|    8 | http://mirrors.aliyun.com/ubuntu | Yes        | No        | 23 hours behind␣
↪| 1.06 MB/s     |
|    9 | http://ftp.sjtu.edu.cn/ubuntu    | Yes        | No        | 23 hours behind␣
↪| 647.2 KB/s    |
|   10 | http://mirrors.yun-idc.com/ubuntu| Yes        | No        | 23 hours behind␣
↪| 526.6 KB/s    |
|   11 | http://mirror.lzu.edu.cn/ubuntu  | Yes        | No        | 23 hours behind␣
↪| 210.99 KB/s   |
|   12 | http://mirrors.ustc.edu.cn/ubuntu| Yes        | Yes       | 8 hours behind  ␣
↪| 455.02 KB/s   |
|   13 | http://mirrors.sohu.com/ubuntu   | No         | No        | Unknown         ␣
↪| 90.28 bytes/s |
-------------------------------------------------------------------------------------------
↪---------------
```

Automatic switching between mirrors The main mirror configured in `/etc/apt/sources.list` can be changed with a single command. The new (to be configured) mirror can be selected automatically or configured explicitly by the user.

Robust package list updating Several apt-get subcommands can fail if the current mirror is being updated (see *issues with mirror updates*) and *apt-smart* tries to work around this by wrapping `apt-get update` to retry on failures and automatically switch to a different mirror when it looks like the current mirror is being updated (because I've seen such updates take more than 15 minutes and it's not always acceptable to wait for so long, especially in automated solutions).

1.1.3 Status

On the one hand the *apt-smart* package was developed based on quite a few years of experience in using apt-get on Debian and Ubuntu systems. On the other hand the Python package itself is relatively new: it was developed and published in Sep 2019. As such:

> **Warning:** Until *apt-smart* has been rigorously tested I consider it a proof of concept (beta software) so if it corrupts your system you can't complain that you weren't warned! The worst that can happen (assuming you trust my judgement ;-) is that `/etc/apt/sources.list` is corrupted however a backup copy is made before any changes are applied, so I don't see how this can result in irreversible corruption.

I'm working on an automated test suite but at the moment I'm still a bit fuzzy on how to create representative tests for the error handling code paths (also, writing a decent test suite requires a significant chunk of time :-).

1.1.4 Installation

The *apt-smart* package is available on PyPI which means installation should be as simple as (paste all below commands together into terminal):

```
sudo apt update
sudo apt install python-pip python-setuptools python-wheel -y  # install python-pip␣
↪and so on without asking
pip install --user apt-smart  # --user flag means install to per user site-packages␣
↪directory(see below)
echo "export PATH=\$(python -c 'import site; print(site.USER_BASE + \"/bin\")'):\$PATH
↪" >> ~/.bashrc
source ~/.bashrc  # set per user site-packages directory to PATH
```

There's actually a multitude of ways to install Python packages (e.g. the per user site-packages directory, virtual environments or just installing system wide) and I have no intention of getting into that discussion here, so if this intimidates you then read up on your options before returning to these instructions ;-).

If a new version of apt-smart has been released, you can upgrade it via:

```
pip install --user apt-smart --upgrade
```

Note. `apt-smart` is a *helper* for the `apt` tool. It is **NOT** a *replacement* for `apt` (or for `apt-get`). So, `apt-smart` should *not* be run *instead* of either of those commands. Nor should `apt-smart` be run with `sudo` or via `su`; if `apt-smart` happens to need root privilege in order for it to continue (in order that it may, for example, change `sources.list`), then it will prompt for a password.

1.1.5 Usage

There are two ways to use the *apt-smart* package: As the command line program `apt-smart` and as a Python API. For details about the Python API please refer to the API documentation available on Read the Docs. The command line interface is described below.

Usage: *apt-smart [OPTIONS]*

The apt-smart program automates robust apt-get mirror selection for Debian and Ubuntu by enabling discovery of available mirrors, ranking of available mirrors, automatic switching between mirrors and robust package list updating.

Supported options:

Option	Description
`-r`, `--remote-host=SSH_ALIAS`	Operate on a remote system instead of the local system. The `SSH_ALIAS` argument gives the SSH alias of the remote host. It is assumed that the remote account has root privileges or password-less sudo access.
`-f`, `--find-current-mirror`	Determine the main mirror that is currently configured in /etc/apt/sources.list and report its URL on standard output.
`-F`, `--file-to-read=local_file_containing_custom_mirrors`	Read a local absolute path (path and filename must NOT contain whitespace) file containing custom mirror URLs (one URL per line) to add custom mirrors to rank.
`-b`, `--find-best-mirror`	Discover available mirrors, rank them, select the best one and report its URL on standard output.
`-l`, `--list-mirrors`	List available (ranked) mirrors on the terminal in a human readable format.
`-L`, `--url-char-len=int`	An integer to specify the length of chars in mirrors' URL to display when using `--list-mirrors`, default is 34
`-c`, `--change-mirror=MIRROR_URL`	Update /etc/apt/sources.list to use the given `MIRROR_URL`.
`-a`, `--auto-change-mirror`	Discover available mirrors, rank the mirrors by connection speed and update status and update /etc/apt/sources.list to use the best available mirror.
`-u`, `--update`, `--update-package-lists`	Update the package lists using "apt-get update", retrying on failure and automatically switch to a different mirror when it looks like the current mirror is being updated.
`-U`, `--ubuntu`	Ubuntu mode for Linux Mint to deal with upstream Ubuntu mirror instead of Linux Mint mirror. e.g. `--auto-change-mirror --ubuntu` will auto-change Linux Mint's upstream Ubuntu mirror
`-x`, `--exclude=PATTERN`	Add a pattern to the mirror selection blacklist. `PATTERN` is expected to be a shell pattern (containing wild cards like "?" and "*") that is matched against the full URL of each mirror.
`-v`, `--verbose`	Increase logging verbosity (can be repeated).
`-V`, `--version`	Show version number and Python version.
`-R`, `--create-chroot=local_dir_absolute_path`	Create chroot with the best mirror in a local directory with absolute_path
`-q`, `--quiet`	Decrease logging verbosity (can be repeated).
`-h`, `--help`	Show this message and exit. **Note: since apt-smart uses *urlopen* method in The Python Standard Library,** you can set Environment Variables to make apt-smart connect via HTTP proxy, e.g. in terminal type: export {http,https,ftp}_proxy='http://user:password@myproxy.com:1080' These will not persist however (no longer active after you close the terminal), so you may wish to add the line to your ~/.bashrc

1.1.6 Issues with mirror updates

The most frequent failure that we run into is `apt-get update` crapping out with 'hash sum mismatch' errors (see also Debian bug #624122). When this happens a file called `Archive-Update-in-Progress-*` can sometimes

be found on the index page of the mirror that is being used (see also Debian bug #110837). I've seen these situations last for more than 15 minutes.

My working theory about these 'hash sum mismatch' errors is that they are caused by the fact that mirror updates aren't atomic, apparently causing `apt-get update` to download a package list whose datafiles aren't consistent with each other. If this assumption proves to be correct (and also assuming that different mirrors are updated at different times :-) then the command `apt-smart --update-package-lists` should work around this annoying failure mode (by automatically switching to a different mirror when 'hash sum mismatch' errors are encountered).

Publishing *apt-smart* to the world is my attempt to contribute to this situation instead of complaining in bug trackers (see above) where no robust and automated solution is emerging (at the time of writing). Who knows, maybe some day these issues will be resolved by moving logic similar to what I've implemented here into `apt-get` itself. Of course it would also help if mirror updates were atomic...

1.1.7 Contact

The latest version of *apt-smart* is available on PyPI and GitHub. The documentation is hosted on Read the Docs and includes a changelog. For bug reports please create an issue on GitHub.

1.1.8 License

This software is licensed under the MIT license.

CHAPTER 2

API documentation

The following API documentation is automatically generated from the source code:

2.1 API documentation

The following documentation is based on the source code of version 7.1.3 of the *apt-smart* package. The following modules are available:

2.1.1 apt_smart

Automated, robust `apt-get` mirror selection for Debian and Ubuntu.

The main entry point for this module is the *AptMirrorUpdater* class, so if you don't know where to start that would be a good place :-). You can also take a look at the source code of the *apt_smart.cli* module for an example that uses the *AptMirrorUpdater* class.

apt_smart.**SOURCES_LIST_ENCODING = 'UTF-8'**
: The text encoding of `main_sources_list` (a string).

apt_smart.**MAX_MIRRORS = 50**
: A sane default value for *AptMirrorUpdater.max_mirrors*.

apt_smart.**URL_CHAR_LEN = 34**
A default value for *AptMirrorUpdater.url_char_len*.

apt_smart.**LAST_UPDATED_DEFAULT = 2419200**
A default, pessimistic *last_updated* value (a number).

class apt_smart.**AptMirrorUpdater**(**kw*)
Python API for the *apt-smart* program.

repr_properties = ('architecture', 'backend', 'blacklist', 'concurrency', 'context', '
Override the list of properties included in repr() output (a tuple of strings).

The PropertyManager superclass defines a __repr__() method that includes the values of computed properties in its output.

In the case of *apt-smart* this behavior would trigger external command execution and (lots of) HTTP calls, sometimes with unintended side effects, namely infinite recursion.

By setting *repr_properties* to a list of "safe" properties this problematic behavior can be avoided.

architecture
The name of the Debian package architecture (a string like 'i386' or 'amd64').

The package architecture is used to detect whether Debian LTS status applies to the given system (the Debian LTS team supports a specific subset of package architectures).

Note: The *architecture* property is a mutable_property. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use del or delattr().

---

available_mirrors
A list of *CandidateMirror* objects (ordered from best to worst)

Note: The *available_mirrors* property is a cached_property. This property's value is computed once (the first time it is accessed) and the result is cached. To clear the cached value you can use del or delattr().

---

backend
The backend module whose name matches *distributor_id*.

Raises EnvironmentError when no matching backend module is available.

Note: The *backend* property is a cached_property. This property's value is computed once (the first time it is accessed) and the result is cached. To clear the cached value you can use del or delattr().

---

best_mirror
The URL of the first mirror in *ranked_mirrors* (a string).

This is a shortcut for using *ranked_mirrors* to select the best mirror from *available_mirrors*, falling back to the old releases URL when *release_is_eol* is True.

Note: The *best_mirror* property is a cached_property. This property's value is computed once (the first time it is accessed) and the result is cached. To clear the cached value you can use del or

delattr().

blacklist
A set of strings with fnmatch patterns (defaults to an empty set).

When *available_mirrors* encounters a mirror whose URL matches one of the patterns in *blacklist* the mirror will be ignored.

Note: The *blacklist* property is a cached_property. This property's value is computed once (the first time it is accessed) and the result is cached. To clear the cached value you can use del or delattr().

concurrency
The number of concurrent HTTP connections allowed while ranking mirrors (a number).

The value of this property defaults to the value computed by *get_default_concurrency()*.

Note: The *concurrency* property is a mutable_property. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use del or delattr().

context
An execution context created using executor.contexts.

The value of this property defaults to a LocalContext object.

Note: The *context* property is a custom_property. You can change the value of this property using normal attribute assignment syntax. This property's value is computed once (the first time it is accessed) and the result is cached. To clear the cached value you can use del or delattr().

current_mirror
The URL of the main mirror in use in *main_sources_list* (a string).

The *current_mirror* property's value is computed using *find_current_mirror()*, but can be changed and cached by *distribution_codename()* for Linux Mint's Ubuntu Mode.

Note: The *current_mirror* property is a custom_property. You can change the value of this property using normal attribute assignment syntax. This property's value is computed once (the first time it is accessed) and the result is cached. To clear the cached value you can use del or delattr().

distribution_codename_old
deprecated: The distribution codename (a lowercase string like 'trusty' or 'xenial').

This relies on executor which is not robust to detect codename when neither /etc/lsb-release nor lsb_release command are available, e.g. the official Debian docker image (see https://github.com/xolox/python-executor/issues/17)

The value of this property defaults to the value of the executor.contexts.AbstractContext.distribution_codename property which is the right choice 99% of the time.

Note: The *distribution_codename_old* property is a mutable_property. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed)

value you can use `del` or `delattr()`.

distribution_codename
The distribution codename (a lowercase string like 'trusty' or 'xenial')

The value of this property is determined using APT sources.list and should be more robust. Similar to *find_current_mirror()* but return token[2] instead. Also refer code of `coerce_release()`.

Note: The *distribution_codename* property is a `custom_property`. You can change the value of this property using normal attribute assignment syntax. This property's value is computed once (the first time it is accessed) and the result is cached. To clear the cached value you can use `del` or `delattr()`.

distributor_id
The distributor ID (a lowercase string like 'debian' or 'ubuntu').

The default value of this property is based on the *distributor_id* property of *release* (which in turn is based on *distribution_codename*).

Because Debian and Ubuntu code names are unambiguous this means that in practice you can provide a value for *distribution_codename* and omit *distributor_id* and everything should be fine.

Note: The *distributor_id* property is a `custom_property`. You can change the value of this property using normal attribute assignment syntax. This property's value is computed once (the first time it is accessed) and the result is cached. To clear the cached value you can use `del` or `delattr()`.

main_sources_list
The absolute pathname of the list of configured APT data sources (a string).

For new version of Linux Mint, main_sources_list is: /etc/apt/sources.list.d/official-package-repositories.list

Note: The *main_sources_list* property is a `cached_property`. This property's value is computed once (the first time it is accessed) and the result is cached. To clear the cached value you can use `del` or `delattr()`.

max_mirrors
Limits the number of mirrors to rank (a number, defaults to *MAX_MIRRORS*).

Note: The *max_mirrors* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

url_char_len
The length of chars in mirrors' URL to display(a number, defaults to *URL_CHAR_LEN*)

Specify the length of chars in mirrors' URL to display when using –list-mirrors

Note: The *url_char_len* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

ubuntu_mode
For Linux Mint, deal with upstream Ubuntu mirror instead of Linux Mint mirror if True

Default is False, can be set True via -U, –ubuntu flag

Note: The *ubuntu_mode* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

old_releases_url
The URL of the mirror that serves old releases for this *backend* (a string).

Note: The *old_releases_url* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

base_url
The actual official base URL according to *BASE_URL*

Note: The *base_url* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

base_last_updated
The Unix timestamp to determine which mirrors are up-to-date (an int)

The value of this property is gotten from *base_url*'s update date as minuend

---

Note: The *base_last_updated* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

ranked_mirrors
A list of *CandidateMirror* objects (ordered from best to worst).

The value of this property is computed by concurrently testing the mirrors in *available_mirrors* for the following details:

- availability (*is_available*)
- connection speed (*bandwidth*)
- update status (*is_updating*)

The number of mirrors to test is limited to *max_mirrors* and you can change the number of simultaneous HTTP connections allowed by setting *concurrency*.

Note: The *ranked_mirrors* property is a `cached_property`. This property's value is computed once (the first time it is accessed) and the result is cached. To clear the cached value you can use `del` or `delattr()`.

---

release
A *Release* object corresponding to *distributor_id* and *distribution_codename*.

Note: The *release* property is a `cached_property`. This property's value is computed once (the first time it is accessed) and the result is cached. To clear the cached value you can use `del` or `delattr()`.

---

release_is_eol
`True` if the release is EOL (end of life), `False` otherwise.

There are three ways in which the value of this property can be computed:

- When available, the first of the following EOL dates will be compared against the current date to determine whether the release is EOL:
  - If the *backend* module contains a `get_eol_date()` function (only the *debian* module does at the time of writing) then it is called and if it returns a number, this number is the EOL date for the release.

    This function was added to enable apt-smart backend modules to override the default EOL dates, more specifically to respect the Debian LTS release schedule (see also issue #5).
  - Otherwise the *eol_date* of *release* is used.
- As a fall back *validate_mirror()* is used to check whether *security_url* results in *MirrorStatus.MAYBE_EOL*.

Note: The *release_is_eol* property is a `cached_property`. This property's value is computed once (the first time it is accessed) and the result is cached. To clear the cached value you can use `del` or `delattr()`.

---

security_url
The URL of the mirror that serves security updates for this *backend* (a string).

Note: The *security_url* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

stable_mirror
A mirror URL that is stable for the given execution context (a string).

The value of this property defaults to the value of *current_mirror*, however if the current mirror can't be determined or is deemed inappropriate by *validate_mirror()* then *best_mirror* will be used instead.

This provides a stable mirror selection algorithm which is useful because switching mirrors causes `apt-get update` to unconditionally download all package lists and this takes a lot of time so should it be avoided when unnecessary.

Note: The *stable_mirror* property is a `cached_property`. This property's value is computed once (the first time it is accessed) and the result is cached. To clear the cached value you can use `del` or `delattr()`.

---

validated_mirrors
Dictionary of validated mirrors (used by *validate_mirror()*).

Note: The *validated_mirrors* property is a `cached_property`. This property's value is computed once (the first time it is accessed) and the result is cached. To clear the cached value you can use `del` or `delattr()`.

---

custom_mirror_file_path
The local custom mirror file's absolute path, can be set by *-F* flag

---

Note: The *custom_mirror_file_path* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

read_custom_mirror_file
Read a file containing custom mirror URLs (one URL per line) to add custom mirrors to rank.

Parameters **file_to_read** – The local file's absolute path

Returns A set of mirrors read from file

Note: The *read_custom_mirror_file* property is a `cached_property`. This property's value is computed once (the first time it is accessed) and the result is cached. To clear the cached value you can use `del` or `delattr()`.

---

change_mirror(*new_mirror=None*, *update=True*)
Change the main mirror in use in *main_sources_list*.

Parameters

- **new_mirror** – The URL of the new mirror (a string, defaults to *best_mirror*).
- **update** – Whether an `apt-get update` should be run after changing the mirror (a boolean, defaults to `True`).

clear_package_lists()
Clear the package list cache by removing the files under `/var/lib/apt/lists`.

create_chroot(*directory*, *codename=None*, *arch=None*)
Bootstrap a basic Debian or Ubuntu system using debootstrap.

Parameters

- **directory** – The pathname of the target directory (a string).
- **codename** – The codename of the target (a string).
- **arch** – The target architecture (a string or `None`).

Returns A `ChangeRootContext` object.

If *directory* already exists and isn't empty then it is assumed that the chroot has already been created and debootstrap won't be run. Before this method returns it changes *context* to the chroot.

dumb_update(**args*)
Update the system's package lists (by running `apt-get update`).

Parameters **args** – Command line arguments to `apt-get update` (zero or more strings).

The *dumb_update()* method doesn't do any error handling or retrying, if that's what you're looking for then you need *smart_update()* instead.

generate_sources_list(***options*)
Generate the contents of `/etc/apt/sources.list`.

If no *mirror_url* keyword argument is given then *stable_mirror* is used as a default.

Please refer to the documentation of the Debian (*apt_smart.backends.debian.generate_sources_list()*) and Ubuntu (*apt_smart.backends.ubuntu.generate_sources_list()*) backend implementations of this method for details on argument handling and the return value.

get_sources_list_options
Get the contents of [options] in *main_sources_list*.

[options] can be set into sources.list, e.g. deb [arch=amd64] http://mymirror/ubuntu bionic main restricted see details at https://manpages.debian.org/jessie/apt/sources.list.5.en.html The [options] is often not considered and breaks parsing in many projects, see https://github.com/jblakeman/apt-select/issues/54 We begin to deal with the [options] by stripping it from sources.list, and then get it back when generating new sources.list

---

Note: The *get_sources_list_options* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

get_sources_list()
Get the contents of *main_sources_list*.

Returns A Unicode string.

This code currently assumes that the `sources.list` file is encoded using *SOURCES_LIST_ENCODING*. I'm not actually sure if this is correct because I haven't been able to find a formal specification! Feedback is welcome :-). This code strips [options] from sources.list, stores it in *get_sources_list_options*

ignore_mirror(*pattern*)
Add a pattern to the mirror discovery *blacklist*.

Parameters **pattern** – A shell pattern (containing wild cards like `?` and `*`) that is matched against the full URL of each mirror.

When a pattern is added to the blacklist any previously cached values of *available_mirrors*, *best_mirror*, *ranked_mirrors* and *stable_mirror* are cleared. This makes sure that mirrors blacklisted after mirror discovery has already run are ignored.

install_sources_list(*contents*)
Install a new `/etc/apt/sources.list` file.

Parameters **contents** – The new contents of the sources list (a Unicode string). You can generate a suitable value using the *generate_sources_list()* method.

smart_update(**args*, ***kw*)
Update the system's package lists (switching mirrors if necessary).

Parameters

- **args** – Command line arguments to `apt-get update` (zero or more strings).
- **max_attempts** – The maximum number of attempts at successfully updating the system's package lists (an integer, defaults to 10).

- **switch_mirrors** – `True` if we're allowed to switch mirrors on 'hash sum mismatch' errors, `False` otherwise.

Raises If updating of the package lists fails 10 consecutive times (*max_attempts*) an exception is raised.

While *dumb_update()* simply runs `apt-get update` the *smart_update()* function works quite differently:

- First the system's package lists are updated using *dumb_update()*. If this is successful we're done.
- If the update fails we check the command's output for the phrase 'hash sum mismatch'. If we find this phrase we assume that the current mirror is faulty and switch to another one.
- Failing `apt-get update` runs are retried up to *max_attempts*.

validate_mirror(*mirror_url*)

Make sure a mirror serves *distribution_codename*.

Parameters **mirror_url** – The base URL of the mirror (a string).

Returns One of the values in the *MirrorStatus* enumeration.

class apt_smart.**CandidateMirror**(***kw*)

A candidate mirror groups a mirror URL with its availability and performance metrics.

bandwidth

The bytes per second achieved while fetching *release_gpg_url* (a number or `None`).

The value of this property is computed based on the values of *release_gpg_contents* and *release_gpg_latency*.

Note: The *bandwidth* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

archive_update_in_progress_url

The URL of the file whose existence indicates that the mirror is being updated (a string).

The value of this property is computed based on the value of *mirror_url*.

Note: The *archive_update_in_progress_url* property is a `lazy_property`. This property's value is computed once (the first time it is accessed) and the result is cached.

---

mirror_url

The base URL of the mirror (a string).

Note: The *mirror_url* property is a `key_property`. You are required to provide a value for this property by calling the constructor of the class that defines the property with a keyword argument named *mirror_url* (unless a custom constructor is defined, in this case please refer to the documentation of that constructor). Once this property has been assigned a value you are not allowed to assign a new value to the property.

---

is_available

`True` if *release_gpg_contents* contains the expected data, `False` otherwise.

The value of this property is computed by checking whether *release_gpg_contents* contains the expected data. This may seem like a rather obscure way of validating a mirror, but it was specifically chosen to detect all sorts of ways in which mirrors can be broken:

- Webservers with a broken configuration that return an error page for all URLs.
- Mirrors whose domain name registration has expired, where the domain is now being squatted and returns HTTP 200 OK responses for all URLs (whether they "exist" or not).

Note: The *is_available* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

is_updating

`True` if the mirror is being updated, `False` otherwise.

Note: The *is_updating* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

last_updated

The time in seconds since the most recent mirror update (a number or `None`).

Note: The *last_updated* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

release_gpg_contents

The contents downloaded from *release_gpg_url* (a string or `None`).

By downloading the file available at *release_gpg_url* and setting *release_gpg_contents* and *release_gpg_latency* you enable the *bandwidth* and *is_available* properties to be computed.

Note: The *release_gpg_contents* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

release_gpg_latency

The time it took to download *release_gpg_url* (a number or `None`).

By downloading the file available at *release_gpg_url* and setting *release_gpg_contents* and *release_gpg_latency* you enable the *bandwidth* and *is_available* properties to be computed.

Note: The *release_gpg_latency* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

release_gpg_url

The URL of the `Release` file that will be used to test the mirror (a string or `None`).

The value of this property is based on *mirror_url* and the *distribution_codename* property of the *updater* object.

Note: The *release_gpg_url* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

sort_key
A tuple that can be used to sort the mirror by its availability/performance metrics.

The tuple created by this property contains four numbers in the following order:

1. The number 1 when *is_available* is `True` or the number 0 when *is_available* is `False` (because most importantly a mirror must be available).
2. The number 0 when *is_updating* is `True` or the number 1 when *is_updating* is `False` (because being updated at this very moment is *bad*).
3. The negated value of *last_updated* (because the lower *last_updated* is, the better). If *last_updated* is `None` then *LAST_UPDATED_DEFAULT* is used instead.
4. The value of *bandwidth* (because the higher *bandwidth* is, the better).

By sorting *CandidateMirror* objects on these tuples in ascending order, the last mirror in the sorted results will be the "most suitable mirror" (given the available information).

Note: The *sort_key* property is a `mutable_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

updater
A reference to the *AptMirrorUpdater* object that created the candidate.

Note: The *updater* property is a `custom_property`. You can change the value of this property using normal attribute assignment syntax. To reset it to its default (computed) value you can use `del` or `delattr()`.

---

class apt_smart.**MirrorStatus**
Enumeration for mirror statuses determined by *AptMirrorUpdater.validate_mirror()*.

AVAILABLE = 1
The mirror is accepting connections and serving the expected content.

MAYBE_EOL = 2
The mirror is serving HTTP 404 "Not Found" responses instead of the expected content.

UNAVAILABLE = 3
The mirror is not accepting connections or not serving the expected content.

apt_smart.**find_current_mirror**(*sources_list*)
Find the URL of the main mirror that is currently in use by `apt-get`.

Parameters **sources_list** – The contents of apt's package resource list, e.g. the contents of `main_sources_list` (a string).

Returns The URL of the main mirror in use (a string).

Raises If the main mirror can't be determined `EnvironmentError` is raised.

The main mirror is determined by looking for the first `deb` or `deb-src` directive in apt's package resource list whose URL uses the HTTP or FTP scheme and whose components contain `main`.

`apt_smart.`**`mirrors_are_equal`**(*a*, *b*)
Check whether two mirror URLS are equal.

Parameters

- **a** – The first mirror URL (a string).
- **b** – The second mirror URL (a string).

Returns `True` if the mirror URLs are equal, `False` otherwise.

`apt_smart.`**`normalize_mirror_url`**(*url*)
Normalize a mirror URL so it can be compared using string equality comparison.

Parameters **url** – The mirror URL to normalize (a string).

Returns The normalized mirror URL (a string).

2.1.2 apt_smart.backends.debian

Discovery of Debian package archive mirrors.

Here are references to some of the material that I've needed to consult while working on this module:

- Notes about sources.list on the Debian wiki
- The Debian backports webpages
- Documentation about the "proposed-updates" mechanism

`apt_smart.backends.debian.`**`LTS_ARCHITECTURES = ('i386', 'amd64', 'armel', 'armhf')`**
The names of the architectures supported by the Debian LTS team (a tuple of strings).

`apt_smart.backends.debian.`**`LTS_RELEASES = {'jessie': 1593468000, 'stretch': 1656540000}`**
A dictionary with Debian LTS releases and their EOL dates.

This is needed because distro-info-data doesn't contain information about Debian LTS releases but nevertheless `archive.debian.org` doesn't adopt a release until the LTS status expires (this was originally reported in issue #5).

`apt_smart.backends.debian.`**`MIRRORS_URL = 'https://www.debian.org/mirror/list'`**
The URL of the HTML page listing all primary Debian mirrors (a string).

`apt_smart.backends.debian.`**`SECURITY_URL = 'http://security.debian.org/'`**
The base URL of the Debian mirror with security updates (a string).

`apt_smart.backends.debian.`**`OLD_RELEASES_URL = 'http://archive.debian.org/debian-archive/deb`**
The URL where EOL (end of life) Debian releases are hosted (a string).

`apt_smart.backends.debian.`**`BASE_URL = 'http://ftp.debian.org/debian/dists/codename-updates/`**
The URL where official repo treated as base are hosted (a string). The Release file contains *Date:* which can be gotten as *`base_last_updated`* to determine which mirrors are up-to-date

`apt_smart.backends.debian.`**`DEFAULT_SUITES = ('release', 'security', 'updates')`**
A tuple of strings with the Debian suites that are enabled by default.

`apt_smart.backends.debian.`**`VALID_COMPONENTS = ('main', 'contrib', 'non-free')`**
A tuple of strings with the names of the components available in the Debian package repositories.

apt_smart.backends.debian.**VALID_SUITES = ('release', 'security', 'updates', 'backports', 'p**
A tuple of strings with the names of the suites available in the Debian package repositories.

apt_smart.backends.debian.**discover_mirrors**()
Discover available Debian mirrors by querying *MIRRORS_URL*.

Returns A set of *CandidateMirror* objects that have their *mirror_url* property set.

Raises If no mirrors are discovered an exception is raised.

An example run:

```
>>> from apt_smart.backends.debian import discover_mirrors
>>> from pprint import pprint
>>> pprint(discover_mirrors())
set([CandidateMirror(mirror_url='http://ftp.at.debian.org/debian/'),
     CandidateMirror(mirror_url='http://ftp.au.debian.org/debian/'),
     CandidateMirror(mirror_url='http://ftp.be.debian.org/debian/'),
     CandidateMirror(mirror_url='http://ftp.bg.debian.org/debian/'),
     CandidateMirror(mirror_url='http://ftp.br.debian.org/debian/'),
     CandidateMirror(mirror_url='http://ftp.by.debian.org/debian/'),
     CandidateMirror(mirror_url='http://ftp.ca.debian.org/debian/'),
     CandidateMirror(mirror_url='http://ftp.ch.debian.org/debian/'),
     CandidateMirror(mirror_url='http://ftp.cn.debian.org/debian/'),
     CandidateMirror(mirror_url='http://ftp.cz.debian.org/debian/'),
     ...])
```

apt_smart.backends.debian.**generate_sources_list**(*mirror_url*, *codename*, *suites=('release', 'security', 'updates')*, *components=('main', 'contrib', 'non-free')*, *enable_sources=False*)
Generate the contents of `/etc/apt/sources.list` for a Debian system.

Parameters

- **mirror_url** – The base URL of the mirror (a string).
- **codename** – The codename of a Debian release (a string like 'wheezy' or 'jessie') or a Debian release class (a string like 'stable', 'testing', etc).
- **suites** – An iterable of strings (defaults to *DEFAULT_SUITES*, refer to *VALID_SUITES* for details).
- **components** – An iterable of strings (refer to *VALID_COMPONENTS* for details).
- **enable_sources** – `True` to include `deb-src` entries, `False` to omit them.

Returns The suggested contents of `/etc/apt/sources.list` (a string).

apt_smart.backends.debian.**get_eol_date**(*updater*)
Override the EOL date for Debian LTS releases.

Parameters **updater** – The *AptMirrorUpdater* object.

Returns The overridden EOL date (a number) or `None`.

2.1.3 apt_smart.backends.ubuntu

Discovery of Ubuntu package archive mirrors.

apt_smart.backends.ubuntu.**MIRRORS_URL = 'https://launchpad.net/ubuntu/+archivemirrors'**
The URL of the HTML page listing official Ubuntu mirrors (a string).

apt_smart.backends.ubuntu.**MIRROR_SELECTION_URL = 'http://mirrors.ubuntu.com/mirrors.txt'**
The URL of a plain text listing of "geographically suitable" mirror URLs (a string).

apt_smart.backends.ubuntu.**OLD_RELEASES_URL = 'http://old-releases.ubuntu.com/ubuntu/'**
The URL where EOL (end of life) Ubuntu releases are hosted (a string).

apt_smart.backends.ubuntu.**SECURITY_URL = 'http://security.ubuntu.com/ubuntu'**
The URL where Ubuntu security updates are hosted (a string).

apt_smart.backends.ubuntu.**BASE_URL = 'http://archive.ubuntu.com/ubuntu/dists/codename-secu**
The URL where official repo treated as base are hosted (a string). The Release file contains *Date:* which can be gotten as *base_last_updated* to determine which mirrors are up-to-date

apt_smart.backends.ubuntu.**DEFAULT_SUITES = ('release', 'updates', 'backports', 'security')**
A tuple of strings with the Ubuntu suites that are enabled by default.

apt_smart.backends.ubuntu.**VALID_COMPONENTS = ('main', 'restricted', 'universe', 'multiverse**
A tuple of strings with the names of the components available in the Ubuntu package repositories.

apt_smart.backends.ubuntu.**VALID_SUITES = ('release', 'security', 'updates', 'backports', 'p**
A tuple of strings with the names of the suites available in the Ubuntu package repositories.

The actual name of the 'release' suite is the codename of the relevant Ubuntu release, while the names of the other suites are formed by concatenating the codename with the suite name (separated by a dash).

As an example to make things more concrete, Ubuntu 16.04 has the following five suites available: `xenial` (this is the release suite), `xenial-security`, `xenial-updates`, `xenial-backports` and `xenial-proposed`.

apt_smart.backends.ubuntu.**discover_mirrors_old**()
Discover available Ubuntu mirrors. (fallback)

Returns A set of *CandidateMirror* objects that have their *mirror_url* property set and may have the *last_updated* property set.

Raises If no mirrors are discovered an exception is raised.

This queries :data:'MIRRORS_URL'to discover available Ubuntu mirrors. Here's an example run:

```
>>> from apt_smart.backends.ubuntu import discover_mirrors_old
>>> from pprint import pprint
>>> pprint(discover_mirrors_old())
set([CandidateMirror(mirror_url='http://archive.ubuntu.com/ubuntu/'),
     CandidateMirror(mirror_url='http://ftp.nluug.nl/os/Linux/distr/ubuntu/'),
     CandidateMirror(mirror_url='http://ftp.snt.utwente.nl/pub/os/linux/ubuntu/'),
     CandidateMirror(mirror_url='http://ftp.tudelft.nl/archive.ubuntu.com/'),
     CandidateMirror(mirror_url='http://mirror.1000mbps.com/ubuntu/'),
     CandidateMirror(mirror_url='http://mirror.amsiohosting.net/archive.ubuntu.
↪com/'),
     CandidateMirror(mirror_url='http://mirror.i3d.net/pub/ubuntu/'),
     CandidateMirror(mirror_url='http://mirror.nforce.com/pub/linux/ubuntu/'),
     CandidateMirror(mirror_url='http://mirror.nl.leaseweb.net/ubuntu/'),
     CandidateMirror(mirror_url='http://mirror.transip.net/ubuntu/ubuntu/'),
     ...])
```

It may be super-slow somewhere (with 100Mbps fibre though) in the world to access launchpad.net (see below), so we have to no longer rely on MIRRORS_URL .

time curl -o/dev/null 'https://launchpad.net/ubuntu/+archivemirrors' % Total % Received % Xferd Average Speed Time Time Time Current

Dload Upload Total Spent Left Speed

100 263k 100 263k 0 0 5316 0 0:00:50 0:00:50 –:–:– 6398

real 0m50.869s user 0m0.045s sys 0m0.039s

But it can be a fallback when MIRROR_SELECTION_URL is down.

apt_smart.backends.ubuntu.**discover_mirrors**()
Discover available Ubuntu mirrors.

Returns A set of *CandidateMirror* objects that have their *mirror_url* property set and may have the *last_updated* property set.

Raises If no mirrors are discovered an exception is raised.

This only queries *MIRROR_SELECTION_URL* to discover available Ubuntu mirrors. Here's an example run: >>> from apt_smart.backends.ubuntu import discover_mirrors >>> from pprint import pprint >>> pprint(discover_mirrors())

apt_smart.backends.ubuntu.**discover_mirror_selection**()
Discover "geographically suitable" Ubuntu mirrors.

apt_smart.backends.ubuntu.**generate_sources_list**(*mirror_url*, *codename*, *suites=('release'*, *'updates'*, *'backports'*, *'security')*, *components=('main'*, *'restricted'*, *'universe'*, *'multiverse')*, *enable_sources=False*)
Generate the contents of `/etc/apt/sources.list` for an Ubuntu system.

Parameters

- **mirror_url** – The base URL of the mirror (a string).
- **codename** – The codename of the Ubuntu release (a string like 'trusty' or 'xenial').
- **suites** – An iterable of strings (defaults to *DEFAULT_SUITES*, refer to *VALID_SUITES* for details).
- **components** – An iterable of strings (refer to *VALID_COMPONENTS* for details).
- **enable_sources** – `True` to include `deb-src` entries, `False` to omit them.

Returns The suggested contents of `/etc/apt/sources.list` (a string).

2.1.4 apt_smart.cli

Usage: *apt-smart [OPTIONS]*

The apt-smart program automates robust apt-get mirror selection for Debian and Ubuntu by enabling discovery of available mirrors, ranking of available mirrors, automatic switching between mirrors and robust package list updating.

Supported options:

Option	Description
-r, --remote-host=SSH_ALIAS	Operate on a remote system instead of the local system. The SSH_ALIAS argument gives the SSH alias of the remote host. It is assumed that the remote account has root privileges or password-less sudo access.
-f, --find-current-mirror	Determine the main mirror that is currently configured in /etc/apt/sources.list and report its URL on standard output.
-F, --file-to-read=local_file_containing_custom	Read a local absolute path (path and filename must NOT contain whitespace) file containing custom mirror URLs (one URL per line) to add custom mirrors to rank.
-b, --find-best-mirror	Discover available mirrors, rank them, select the best one and report its URL on standard output.
-l, --list-mirrors	List available (ranked) mirrors on the terminal in a human readable format.
-L, --url-char-len=int	An integer to specify the length of chars in mirrors' URL to display when using --list-mirrors, default is 34
-c, --change-mirror=MIRROR_URL	Update /etc/apt/sources.list to use the given MIRROR_URL.
-a, --auto-change-mirror	Discover available mirrors, rank the mirrors by connection speed and update status and update /etc/apt/sources.list to use the best available mirror.
-u, --update, --update-package-lists	Update the package lists using "apt-get update", retrying on failure and automatically switch to a different mirror when it looks like the current mirror is being updated.
-U, --ubuntu	Ubuntu mode for Linux Mint to deal with upstream Ubuntu mirror instead of Linux Mint mirror. e.g. --auto-change-mirror --ubuntu will auto-change Linux Mint's upstream Ubuntu mirror
-x, --exclude=PATTERN	Add a pattern to the mirror selection blacklist. PATTERN is expected to be a shell pattern (containing wild cards like "?" and "*") that is matched against the full URL of each mirror.
-v, --verbose	Increase logging verbosity (can be repeated).
-V, --version	Show version number and Python version.
-R, --create-chroot=local_dir_absolute_path	Create chroot with the best mirror in a local directory with absolute_path
-C, --codename=codename	Must use with -R , create chroot with a codename of Ubuntu or Debian, e.g. bionic, buster
-q, --quiet	Decrease logging verbosity (can be repeated).
-h, --help	Show this message and exit. **Note: since apt-smart uses *urlopen* method in The Python Standard Library,** you can set Environment Variables to make apt-smart connect via HTTP proxy, e.g. in terminal type: export {http,https,ftp}_proxy='http://user:password@myproxy.com:1080' These will not persist however (no longer active after you close the terminal), so you may wish to add the line to your ~/.bashrc

apt_smart.cli.**main**()
Command line interface for the apt-smart program.

apt_smart.cli.**report_current_mirror**(*updater*)
Print the URL of the currently configured apt-get mirror.

apt_smart.cli.**report_best_mirror**(*updater*)
Print the URL of the "best" mirror.

apt_smart.cli.**report_available_mirrors**(*updater*)

Print the available mirrors to the terminal (in a human friendly format).

2.1.5 apt_smart.http

Simple, robust and concurrent HTTP requests (designed for one very narrow use case).

apt_smart.http.**fetch_url**(*url*, *timeout=10*, *retry=False*, *max_attempts=3*)
Fetch a URL, optionally retrying on failure.

Parameters

- **url** – The URL to fetch (a string).
- **timeout** – The maximum time in seconds that's allowed to pass before the request is aborted (a number, defaults to 10 seconds).
- **retry** – Whether to retry on failure (defaults to `False`).
- **max_attempts** – The maximum number of attempts when retrying is enabled (an integer, defaults to three).

Returns The response body (a byte string).

Raises Any of the following exceptions can be raised:

- *NotFoundError* when the URL returns a 404 status code.
- *InvalidResponseError* when the URL returns a status code that isn't 200.
- `stopit.TimeoutException` when the request takes longer than *timeout* seconds (refer to the stopit documentation for details).
- Any exception raised by Python's standard library in the last attempt (assuming all attempts raise an exception).

apt_smart.http.**fetch_concurrent**(*urls*, *concurrency=None*)
Fetch the given URLs concurrently using `multiprocessing`.

Parameters

- **urls** – An iterable of URLs (strings).
- **concurrency** – Override the concurrency (an integer, defaults to the value computed by *get_default_concurrency()*).

Returns A list of tuples like those returned by *fetch_worker()*.

apt_smart.http.**get_default_concurrency**()
Get the default concurrency for *fetch_concurrent()*.

Returns A positive integer number.

apt_smart.http.**fetch_worker**(*url*)
Fetch the given URL for *fetch_concurrent()*.

Parameters **url** – The URL to fetch (a string).

Returns

A tuple of three values:

1. The URL that was fetched (a string).
2. The data that was fetched (a string or `None`).
3. The number of seconds it took to fetch the URL (a number).

exception apt_smart.http.**InvalidResponseError**
Raised by *fetch_url()* when a URL returns a status code that isn't 200.

exception apt_smart.http.**NotFoundError**
Raised by *fetch_url()* when a URL returns a 404 status code.

2.1.6 apt_smart.releases

Easy to use metadata on Debian and Ubuntu releases.

This module started out with the purpose of reliable end of life (EOL) detection for Debian and Ubuntu releases based on data provided by the distro-info-data package. Since then the need arose to access more of the available metadata and so the `eol` module became the `releases` module.

Debian and Ubuntu releases have an EOL date that marks the end of support for each release. At that date the release stops receiving further (security) updates and some time after package mirrors stop serving the release.

The distro-info-data package contains CSV files with metadata about Debian and Ubuntu releases. This module parses those CSV files to make this metadata available in Python. This enables *apt-smart* to make an informed decision about the following questions:

1. Is a given Debian or Ubuntu release expected to be available on mirrors or will it only be available in the archive of old releases?
2. Is the signing key of a given Ubuntu release expected to be included in the main keyring (*UBUNTU_KEYRING_CURRENT*) or should the keyring with removed keys (*UBUNTU_KEYRING_REMOVED*) be used?

To make it possible to run *apt-smart* without direct access to the CSV files, a copy of the relevant information has been embedded in the source code.

apt_smart.releases.**DISTRO_INFO_DIRECTORY = '/usr/share/distro-info'**
The pathname of the directory with CSV files containing release metadata (a string).

apt_smart.releases.**DEBIAN_KEYRING_CURRENT = '/usr/share/keyrings/debian-archive-keyring.gpg**
The pathname of the main Debian keyring file (a string).

apt_smart.releases.**UBUNTU_KEYRING_CURRENT = '/usr/share/keyrings/ubuntu-archive-keyring.gpg**
The pathname of the main Ubuntu keyring file (a string).

apt_smart.releases.**UBUNTU_KEYRING_REMOVED = '/usr/share/keyrings/ubuntu-archive-removed-key**
The pathname of the Ubuntu keyring file with removed keys (a string).

apt_smart.releases.**coerce_release**(*value*)
Try to coerce the given value to a Debian or Ubuntu release.

Parameters **value** – The value to coerce (a number, a string or a *Release* object).

Returns A *Release* object.

Raises `ValueError` when the given value cannot be coerced to a known release.

The following values can be coerced:

- Numbers and numbers formatted as strings match *Release.version*.
- Strings match *Release.codename* (case insensitive).

> **Warning:** Don't use floating point numbers like 10.04 because their actual value will be something like 10.039999999999999147 which won't match the intended release.

apt_smart.releases.**discover_releases**()
Discover known Debian and Ubuntu releases.

Returns A list of discovered *Release* objects sorted by *distributor_id* and *version*.

The first time this function is called it will try to parse the CSV files in /usr/share/distro-info and merge any releases it finds with the releases embedded into the source code of this module. The result is cached and returned each time the function is called. It's not a problem if the /usr/share/distro-info directory doesn't exist or doesn't contain any *.csv files (it won't cause a warning or error). Of course in this case only the embedded releases will be returned.

apt_smart.releases.**ubuntu_keyring_updated**()
Detect update #1363482 to the ubuntu-keyring package.

Returns True when version 2016.10.27 or newer is installed, False when an older version is installed.

This function checks if the changes discussed in Launchpad bug #1363482 apply to the current system using the dpkg-query --show and dpkg --compare-versions commands. For more details refer to issue #8.

class apt_smart.releases.**Release**(***kw*)
Data class for metadata on Debian and Ubuntu releases.

codename
The long version of *series* (a string).

Note: The *codename* property is a key_property. You are required to provide a value for this property by calling the constructor of the class that defines the property with a keyword argument named *codename* (unless a custom constructor is defined, in this case please refer to the documentation of that constructor). Once this property has been assigned a value you are not allowed to assign a new value to the property.

---

compatible_repository
For Linux Mint, compatible which Ubuntu version's repository

Note: The *compatible_repository* property is a writable_property. You can change the value of this property using normal attribute assignment syntax.

---

created_date
The date on which the release was created (a date object).

Note: The *created_date* property is a required_property. You are required to provide a value for this property by calling the constructor of the class that defines the property with a keyword argument named *created_date* (unless a custom constructor is defined, in this case please refer to the documentation of that constructor). You can change the value of this property using normal attribute assignment syntax.

---

distributor_id
The name of the distributor (a string like debian or ubuntu or linuxmint).

Note: The *distributor_id* property is a key_property. You are required to provide a value for this property by calling the constructor of the class that defines the property with a keyword argument named *distributor_id* (unless a custom constructor is defined, in this case please refer to the documentation

of that constructor). Once this property has been assigned a value you are not allowed to assign a new value to the property.

eol_date
The date on which the desktop release stops being supported (a `date` object).

Note: The *eol_date* property is a `writable_property`. You can change the value of this property using normal attribute assignment syntax.

extended_eol_date
The date on which the server release stops being supported (a `date` object).

Note: The *extended_eol_date* property is a `writable_property`. You can change the value of this property using normal attribute assignment syntax.

is_eol
Whether the release has reached its end-of-life date (a boolean or `None`).

Note: The *is_eol* property is a `lazy_property`. This property's value is computed once (the first time it is accessed) and the result is cached.

is_lts
Whether a release is a long term support release (a boolean).

Note: The *is_lts* property is a `writable_property`. You can change the value of this property using normal attribute assignment syntax.

release_date
The date on which the release was published (a `date` object).

Note: The *release_date* property is a `writable_property`. You can change the value of this property using normal attribute assignment syntax.

series
The short version of *codename* (a string).

Note: The *series* property is a `key_property`. You are required to provide a value for this property by calling the constructor of the class that defines the property with a keyword argument named *series* (unless a custom constructor is defined, in this case please refer to the documentation of that constructor). Once this property has been assigned a value you are not allowed to assign a new value to the property.

version
The version number of the release (a `Decimal` number).

This property has a `Decimal` value to enable proper sorting based on numeric comparison.

Note: The *version* property is a `writable_property`. You can change the value of this property using normal attribute assignment syntax.

keyring_file
The pathname of the keyring with signing keys for this release (a string).

This property exists to work around a bug in `debootstrap` which may use the wrong keyring to create Ubuntu chroots, for more details refer to *ubuntu_keyring_updated()*.

Note: The *keyring_file* property is a `lazy_property`. This property's value is computed once (the first time it is accessed) and the result is cached.

__str__()
Render a human friendly representation of a *Release* object.

The result will be something like this:

- Debian 9 (stretch)
- Ubuntu 18.04 (bionic)

CHAPTER 3

Change log

The change log lists notable changes to the project:

3.1 Changelog

The purpose of this document is to list all of the notable changes to this project. The format was inspired by Keep a Changelog. This project adheres to semantic versioning.

3.1.1 Release 7.1.3 (2020-5-31)

- Support 'mirror://' scheme: https://github.com/martin68/apt-smart/issues/3
- Update releases.py bundled Releases by running 'make releases', related https://github.com/martin68/apt-smart/issues/4
- In python2 decode() default encoding is ascii, causing https://github.com/martin68/apt-smart/issues/5 , specify utf-8
- Fix current_mirror in linuxmint's ubuntu mode, causing -U -c 'mirror_url' changed linuxmint's mirror_url instead of ubuntu's

3.1.2 Release 7.1.2 (2019-11-28)

- Support Python 3.8
- Add *-C –codename* flag to create chroot with a distribution codename.
- Blacklist BASE_URL mirror if matches blacklist pattern, this helps when BASE_URL (official) mirror is the only up-to-date one and you find it so slow that you'd like to blacklist it.

3.1.3 Release 7.1.1 (2019-11-04)

- For Linux Mint, backup official-package-repositories.list to backup_dir: backup_by_apt-smart
- In Readme, add install commands for Linux Mint and a note about run with sudo

3.1.4 Release 7.1.0 (2019-11-01)

- Add support for Linux Mint
- Add -U, –ubuntu to opt in Ubuntu mode for Linux Mint to deal with upstream Ubuntu mirror instead of Linux Mint mirror. e.g. –auto-change-mirror –ubuntu will auto-change Linux Mint's upstream Ubuntu mirror

3.1.5 Release 7.0.7 (2019-9-30)

- Fix install_sources_list() for Python 3
- Fix –change-mirror
- fix Travis CI *io.UnsupportedOperation:fileno* error by changing the way to run test cases containing smart_update()
- More test cases

3.1.6 Release 7.0.6 (2019-9-25)

- Readme & help about proxy setting
- Deal with the [options] in sources.list by stripping it from sources.list, and then get it back when generating new sources.list , fix https://github.com/jblakeman/apt-select/issues/54
- Add a warning: custom mirror file's path and filename must NOT contain whitespace
- Add -R, –create-chroot=local_dir_absolute_path to create chroot with the best mirror in a local directory with absolute_path
- More test cases

3.1.7 Release 7.0.5 (2019-9-21)

- Add -F, –file-to-read=local_file_absolute_path (path and filename must NOT contain whitespace) to read a local absolute path file containing custom mirror URLs (one URL per line) to add custom mirrors to rank. So now you can use e.g. *-l -F ~/mirrors.txt* options to add some custom mirrors to rank with mirrors in official mirror list.
- Updated BUNDLED_RELEASES in releases.py

3.1.8 Release 7.0.4 (2019-9-20)

- Fix error on EOL release
- Check OLD_RELEASES_URL's MirrorStatus to confirm if it is EOL, to fix https://github.com/xolox/python-apt-mirror-updater/issues/9

3.1.9 Release 7.0.3 (2019-9-19)

- Fix *–url-char-len* option to specify the length of chars in mirrors' URL to display when using *–list-mirrors*, so that now you can use e.g. *-l -L 29* options to narrow down the table of ranked mirrors when you want to paste it to somewhere the table displayed badly.

3.1.10 Release 7.0.2 (2019-9-19)

- Add :attr:url_char_len to specify the length of chars in mirrors' URL to display when using *–list-mirrors*, so that now you can use e.g. *-l -L 29* options to narrow down the table of ranked mirrors when you want to paste it to somewhere the table displayed badly.

3.1.11 Release 7.0.1 (2019-9-18)

- Better output format when use *–list-mirrors*

3.1.12 Release 7.0 (2019-9-15)

- Rename the project and module to `apt-smart`
- For Ubuntu, new mirrors discovery mechanism: at first it queries `MIRROR_SELECTION_URL`, and `MIRRORS_URL` as fallback.
- For Debian, new mirrors discovery mechanism: get mirrors within the country which the user is in.
- New mechanism of determining whether a mirror is up-to-date: download the InRelease file and parse the `Date` value in it.
- New and more robust `distribution_codename` using APT sources.list
- Enable retry when `fetch_url` is timeout for bad connections.
- Drop Python 2.6 support and add Python 3.7
- Drop max_mirrors limit since we can smartly get mirrors within the user's country.

3.1.13 Release 6.1 (2018-10-19)

- Bug fix for Ubuntu keyring selection that prevented `ubuntu-archive-removed-keys.gpg` from being used.
- Bug fix for `coerce_release()` when given a release number.
- Moved pathnames of Debian and Ubuntu keyring files to constants.
- Added logging to enable debugging of keyring selection process.
- Added proper tests for keyring selection and release coercion.

3.1.14 Release 6.0 (2018-10-14)

Enable the creation of Ubuntu <= 12.04 chroots on Ubuntu >= 17.04 hosts by working around (what I am convinced is) a bug in `debootstrap` which picks the wrong keyring when setting up chroots of old releases. For more information refer to issue #8.

I've bumped the major version number for this release because the highly specific `apt_smart.eol` module changed into the much more generic `apt_smart.releases` module. Also the `release_label` property was removed.

3.1.15 Release 5.2 (2018-10-08)

Use mirrors.ubuntu.com/mirrors.txt without placing our full trust in it like older versions of `apt-smart` did .

Feedback in issue #6 suggested that mirrors.ubuntu.com/mirrors.txt is working properly (again) and should be preferred over scraping Launchpad. However I prefer for `apt-smart` to be a reliable "do what I mean" program and mirrors.ubuntu.com/mirrors.txt has proven to be unreliable in the past (see the discussion in #6). As a compromise I've changed the Ubuntu mirror discovery as follows:

1. Discover Ubuntu mirrors on Launchpad.
2. Try to discover mirrors using mirrors.ubuntu.com/mirrors.txt and iff successful, narrow down the list produced in step 1 based on the URLs reported in step 2.
3. Rank the discovered / narrowed down mirrors and pick the best one.

The reason why I've decided to add this additional complexity is because it has bothered me in the past that Ubuntu mirror discovery was slow and this does help a lot. Also, why not use a service provided by Ubuntu to speed things up?

Unrelated to the use of mirrors.ubuntu.com/mirrors.txt I've also bumped the `executor` requirement (twice) in order to pull in upstream improvements discussed in executor issue #10 and executor issue #15.

3.1.16 Release 5.1 (2018-06-22)

Work on release 5.1 started with the intention of publishing a 5.0.2 bug fix release for the EOL detection of Debian LTS releases reported in #5, however unrelated changes were required to stabilize the test suite. This explains how 5.0.2 became 5.1 .

When I started working on resolving the issue reported in #5 it had been quite a while since the previous release (233 days) and so some technical debt had accumulated in the project, causing the test suite to break. Most significantly, Travis CI switched their workers from Ubuntu 12.04 to 14.04.

Here's a detailed overview of changes:

- Bug fix for EOL detection of Debian LTS releases (reported in #5).
- Bug fix for trivial string matching issue in test suite (caused by a naively written test).
- Bug fix for recursive `repr()` calls potentially causing infinite recursion, depending on logging level (see e.g. build 395421319).
- Updated bundled EOL dates based on distro-info-data available in Ubuntu 18.04.
- Added this changelog to the documentation, including a link in the readme.
- Make sure the `test_gather_eol_dates` test method runs on Travis CI (by installing the distro-info-data package). This exposed a Python 3 incompatibility (in build 395410569) that has since been resolved.
- Include documentation in source distributions (`MANIFEST.in`).
- Silence flake8 complaining about bogus D402 issues.
- Add license='MIT' key to `setup.py` script.
- Bumped copyright to 2018.

3.1.17 Release 5.0.1 (2017-11-01)

Bug fix release for invalid enumeration value (oops).

3.1.18 Release 5.0 (2017-11-01)

Reliable end of life (EOL) detection.

Recently I ran into the issue that the logic to check whether a release is EOL (that works by checking if the security mirror serves a `Release.gpg` file for the release) failed on me. More specifically the following URL existed at the time of writing (2017-11-01) even though Ubuntu 12.04 went EOL back in April:

http://security.ubuntu.com/ubuntu/dists/precise/Release.gpg

At the same time issue #1 and pull request #2 were also indications that the EOL detection was fragile and error prone. This potential fragility had bugged me ever since publishing *apt-smart* and this week I finally finished a more robust and deterministic EOL detection scheme.

This release includes pull requests #2 and #4, fixing issues #1 and #3. Here's a detailed overview of changes:

- Addition: Allow optional arguments to `apt-get update` (#3, #4).
  - I simplified and improved the feature requested in issue #3 and implemented in pull request #4 by switching from an optional list argument to 'star-args' and applying the same calling convention to `smart_update()` as well.
  - This is backwards incompatible with the implementation in pull request #4 (which I merged into the `dev` branch but never published to PyPI) and it's also technically backwards incompatible in the sense that keyword arguments could previously be given to `smart_update()` as positional arguments. This explains why I'm bumping the major version number.
- Bug fix for incorrect marking of EOL when HTTP connections fail (#2).
- Refactoring: Apply timeout handling to HTTP response bodies.
- Refactoring: Distinguish 404 from other HTTP errors:
  - This change enhances `validate_mirror()` by making a distinction between a confirmed HTTP 404 response versus other error conditions which may be of a more transient nature.
  - The goal of this change is to preserve the semantics requested in issue #1 and implemented in pull request #2 without needing the additional HTTP request performed by `can_connect_to_mirror()`.
  - Because `validate_mirror()` previously returned a boolean but now returns an enumeration member this change is technically backwards incompatible, then again `validate_mirror()` isn't specifically intended for callers because it concerns internal logic of apt-smart. I'm nevertheless bumping the major version number.
- Refactoring: Improve HTTP request exception handling:
  - 404 responses and timeouts are no longer subject to retrying.
  - The exception `apt_smart.http.NotFoundError` is now raised on HTTP 404 responses. Other unexpected HTTP response codes raise `apt_smart.http.InvalidResponseError`.
  - The specific distinction between 404 and !200 was made because the 404 response has become significant in checking for EOL status.

3.1.19 Release 4.0 (2017-06-14)

Robust validation of available mirrors (backwards incompatible).

3.1.20 Release 3.1 (2017-06-13)

Made mirror comparison more robust.

3.1.21 Release 3.0 (2017-06-13)

Added Debian archive support (with old releases):

- Addition: Added Debian archive support (old releases).
- Improvement: Don't bother validating archive / old-releases mirror.
- Refactoring: Moved URLs to backend specific modules.

3.1.22 Release 2.1 (2017-06-12)

Restored Python 3 compatibility, improved robustness:

- Improvement: Make the `is_available` and `is_updating` properties of the `CandidateMirror` class more robust.
- Bug fix: I suck at Unicode in Python (most people do :-p).
- Cleanup: Remove unused import from test suite.

3.1.23 Release 2.0 (2017-06-11)

Generation of `sources.list` files and chroot creation.

Detailed overview of changes:

- Addition: Added a simple `debootstrap` wrapper.
- Addition: Programmatic /etc/apt/sources.list generation
- Bug fix for `check_suite_available()`.
- Bug fix: Never apply Ubuntu's old release handling to Debian.
- Bug fix: Never remove `/var/lib/apt/lists/lock file`.
- Improvement: Enable stable mirror selection
- Improvement: Make it possible to override distributor ID and codename
- Improvement: Render interactive spinner during mirror ranking.
- Refactoring: Generalize AptMirrorUpdater initializer (backwards incompatible!)
- Refactoring: Generalize backend module loading
- Refactoring: Modularize `/etc/apt/sources.list` writing.

3.1.24 Release 1.0 (2017-06-08)

Improved Ubuntu mirror discovery, added an automated test suite, and more.

The bump to version 1.0 isn't so much intended to communicate that this is now mature software, it's just that I made several backwards incompatible changes in order to improve the modularity of the code base, make it easier to develop automated tests, maintain platform support, etc :-).

A more detailed overview of (significant) changes:

- Improved Ubuntu mirror discovery (by scraping Launchpad instead).
- Extracted mirror discovery to separate (backend specific) modules.

- Extracted HTTP handling to a separate module.
- Enable Control-C to interrupt concurrent connection tests.
- Expose limit in Python API and command line interface and make limit optional by passing 0.
- Bug fix for Python 3 incompatibility: Stop using `sys.maxint` :-).

3.1.25 Release 0.3.1 (2016-06-29)

Avoid 'nested' smart updates (the old code worked fine but gave confusing output and performed more work than necessary, which bothered me :-).

3.1.26 Release 0.3 (2016-06-29)

Make smart update understand EOL suites

3.1.27 Release 0.2 (2016-06-29)

Bug fix: Replace `security.ubuntu.com` as well.

3.1.28 Release 0.1.2 (2016-06-29)

Bug fix: Explicitly terminate multiprocessing pool.

3.1.29 Release 0.1.1 (2016-03-10)

Initial release (added `MANIFEST.in`).

3.1.30 Release 0.1 (2016-03-10)

Initial commit.

Python Module Index

a

Index

Symbols

A

B

C

D

E

F

G

I

K

L

M

N

O

R

S

U

V