PPA Tools

Launchpad has an API, but it's not very well documented and only has libraries for Python (although since it's a web service you could write your own with curl). I don't want to force myself to use Python for everything PPA-related, I don't want to reinvent the wheel with curl, but I do want a set of tools that can let me perform common functions from my other scripts. The **ppa-tools** command is exactly that.

1. Introduction

This is not a user-friendly way to get access to your PPA, it's mostly written as a library with a set of deterministic outputs. For example, you'll find that most output is on the last line as a space-separated set of values; this is intentional so running **ppa-tools [stuff] | tail -n1 | awk '{print $1 + $2}'** is easy to do. I also don't include things like optional parameters or default values, even though I typically always use this on my own PPA IO log in anonymously when possible and force you to specify the PPA owner and name on the command line. I feel this maximizes usability as a library, minimizes the risk that I need to change the inputs and outputs, and since I'm using the script programmatically anyway I just set the defaults in the master script.

However, **ppa-tools** can be used as a standalone utility just fine. It's a little clumsy and spartan for that purpose, but I do offer help and readable output (the reason for the **tail -n1** is because I do print a header row so you know what you're looking at).

2. Usage

I try to keep the syntax consistent. I try to stick to **ppa-tools command [ppa_owner ppa_name] [ppa_name] [package_name] [package_version] [series]** with a notable exception of the copy command which requires two series.

ppa-tools get_series

Prints a space-separated list of all active series Launchpad supports. This excludes deprecated series but also excludes some future series available via the web UI (for example, as of May 2012 `quantal` is in the web UI but not returned by this command).

```
$ ppa-tools get_series
precise oneiric natty lucid hardy
```

ppa-tools get_pkg_ver ppa_owner ppa_name pkg_name series_name

Prints the version of the source for a particular package. Note that this means that the source has been submitted, but not necessarily that the binaries are done building.

```
$ ppa-tools get_pkg_ver ubuntu-ebower ebower tabletify precise
0.1.4
```

ppa-tools get_pkg_status ppa_owner ppa_name pkg_name series_name

Prints the status of the source for a particular package. Since it's the status of the source, this does not mean that the binaries are available.

```
$ ppa-tools get_pkg_status ubuntu-ebower ebower ppa-tools precise
Published
```

ppa-tools get_bin_status ppa_owner ppa_name pkg_name pkg_version series_name

Prints a two-line output showing the status of the binaries for a particular package and version. The first line is the header so you know which status represents which architecture. The second line is the status.

```
$ ppa-tools get_bin_status ubuntu-ebower ebower ppa-tools 0.1 precise
i386 amd64
Published Published
```

ppa-tools get_pkgs ppa_owner ppa_name

Prints a single line, space separated list of all package names in a given PPA.

```
$ ppa-tools get_pkgs ubuntu-ebower ebower
dbcompile  makemkv-install  mkv-rename  multi-build  ppa-tools  sparkline-gp  tabletify
```

ppa-tools get_pkgs_full ppa_owner ppa_name

Not incredibly programmatically friendly, but a useful data dump. It prints the package name, version, series, architecture, and binary status for everything in the PPA. Since the output is fairly large, the example below is truncated.

```
$ ppa-tools get_pkgs_full ubuntu-ebower ebower |head
pkg_name pkg_version in series architecture bin_status
dbcompile 0.1.2.1 in quantal i386 Published
dbcompile 0.1.2.1 in quantal amd64 Published
dbcompile 0.1.2.1 in precise i386 Published
dbcompile 0.1.2.1 in precise amd64 Published
dbcompile 0.1.2.1 in oneiric i386 Published
dbcompile 0.1.2.1 in oneiric amd64 Published
dbcompile 0.1.2.1 in lucid amd64 Published
dbcompile 0.1.2.1 in lucid i386 Published
dbcompile 0.1.2.1 in natty amd64 Published
```

ppa-tools get_count ppa_owner ppa_name pkg_name

Prints a header row followed by the number of downloads for each architecture. Yes, I cherry-picked the example...

```
$ ppa-tools get_count ubuntu-ebower ebower makemkv-install
i386 amd64
214 388
```

ppa-tools cp_pkg ppa_owner ppa_name pkg_name pkg_name source_series dest_series

This command requires that you allow access to your Launchpad account because it's making a change rather than just reading data, this also means the script could do bad things so consider yourself warned. The first time you use this command it will open a browser window to ask you to generate an authentication token, feel free to only give a small window for access or just give it permanent access if you'd rather not need to log in every time you use it. I use this script on my own PPA and you're welcome to read the code to verify it's not malicious, but there's always the chance I screwed something up or will do so in the future.

This command will copy the binaries from one series to another. PPAs, for some reason, allow you to submit to one series at a time. So your option are to submit to all series at once, using up a slew of build servers, or to wait for a primary series to complete and then copy the binaries over. In my case, the "binaries" are scripts so compiling seems like a waste. The output is not very programmatic, if it didn't throw an exception it worked.

```
$ ppa-tools cp_pkg ubuntu-ebower ebower uping precise quantal
Copying uping 0.2.7.1 from precise to quantal
```

ppa-tools version|--version|-v

Prints the help, the version, and exits. Once again, **tail -n1 | awk '{print $3}'** is your friend.

```
$ ppa-tools version
Usage:
ppa-tools command [options]

Command options (run ppa-tools command to see details):
get_series - prints a list of all active series
get_pkg_ver - prints the version of a package
get_pkg_status - prints the current status of the package
get_bin_status - prints the status of the binaries in a package
get_pkgs - prints a list of the packages in a PPA
get_pkgs_full - prints a detailed list of the packages in a PPA
get_count - prints the number of times a package was downloaded
cp_pkg - copies a package to another series in a PPA
--version|-v - prints the current version of this program

ppa-tools version 0.1
```

3. Installation

Most Ubuntu users should strongly consider using my Launchpad PPA which will help keep your package up-to-date. You can install it with the following commands:

```
sudo add-apt-repository ppa:ubuntu-ebower/ebower
sudo apt-get update
sudo apt-get install ppa-tools
```

If you're not fortunate enough to be able to use this, you can download the packages here:

Table 1. Download links for "ppa-tools"

Distro Type	i386	amd64
Debian	ppa-tools_0.1_i386.deb	ppa-tools_0.1_amd64.deb
RedHat	ppa-tools-0.1-2.i386.rpm	ppa-tools-0.1-2.x86_64.rpm
Other	ppa-tools_0.1.tar.gz	ppa-tools_0.1.tar.gz

3.1. Installing Under Windows

I've made a conscious decision to develop exclusively for Linux, if you use Windows or (heaven help you) OSX you can still run my programs for free. If I developed for Windows, you'd need to hope that Wine works or fork over the money for a Windows install disc. If I developed for OSX you would be legally obligated to buy Mac hardware due to restrictive licensing. Since I develop for Linux, you can obtain Linux with zero additional cost and run it on just about any system known to man.

First, try out Linux. I use Ubuntu (http://www.ubuntu.com) which is pretty user-friendly, but feel free to check out other distros (http://distrowatch.com/dwres.php?resource=major) (or, if you want a more complete list, check out the excellent GNU/Linux Distribution Timeline (http://futurist.se/gldt/)). Fedora is another choice but it uses a different root than Ubuntu (RedHat vs. Debian) so some things will be different. Personally I'd rather be running Slackware, but that's not for someone who needs to read how to install my stuff under Windows. Download a CD and try it out, worst case you'll like it and install it alongside your existing OS or even as your primary OS.

Barring that, try out VirtualBox (http://www.virtualbox.org), installing this will allow you to install Linux within your current OS. This will let you run my programs just fine and try out Linux without making any changes to your existing system.

Finally, Windows users can also try Cygwin (http://www.cygwin.com). It won't let you run Linux programs, but since the majority of my "programs" are really scripts you may be able to get them to work.

A. About Me

My name is Jeff Bower, I'm a technology professional (http://www.linkedin.com/in/jdbower) with more years of experience in the telecommunications industry than I'd care to admit. I tend to post with the username jdbower on various forums, including Komodo Kamado (http://komodokamado.com/forum/), Android Central (http://forum.androidcentral.com/), VirtualBox (http://forums.virtualbox.org/), and MakeMKV (http://www.makemkv.com/forum2/). Writing these documents is a hobby of mine, I hope you find them useful and feel free to browse more at https://www.ebower.com/docs.

I also enjoy cooking, especially outdoors with my Komodo Kamado (http://www.komodokamado.com) and using my Stoker (https://www.rocksbarbque.com). Take a look at my recipes stored at https://www.ebower.com/recipes.

If you've got any questions or feedback please feel free to email me at docs@ebower.com (mailto:docs@ebower.com) or follow me on Google+ (https://profiles.google.com/100268310848930740059) or Twitter (http://twitter.com/jdbower).