



Things I Learned from the AutoPkg Maintainers




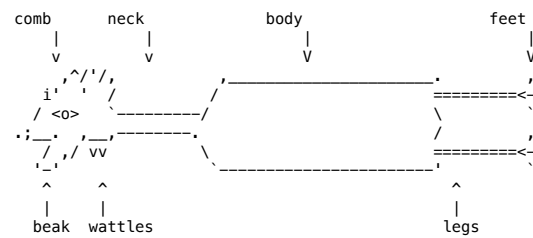
Anthony Reimer

 maclabs.jazzace.ca

 [AnthonyReimer](https://twitter.com/AnthonyReimer)

 [jazzace](https://jazzace.ca)

 [jazzace](https://github.com/jazzace)



<https://tools.ietf.org/html/rfc2321>

[maclabs.jazzace.ca/2020/06/04/
things-i-learned-autopkg](https://maclabs.jazzace.ca/2020/06/04/things-i-learned-autopkg)



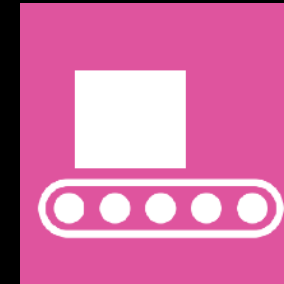
The slides and resources for this presentation are already posted on my blog at the URL shown. I'll show this link again at the end of the presentation. If you have any questions *during* the presentation, enter them using the Q&A function as soon as you think of them. The moderator will collect them and I'll respond at the end of the presentation. Better yet, if you are reminded of something *you've* learned from the AutoPkg developers and maintainers that seems apropos, add those comments in the Q&A as well.

What I'm *Not* Discussing

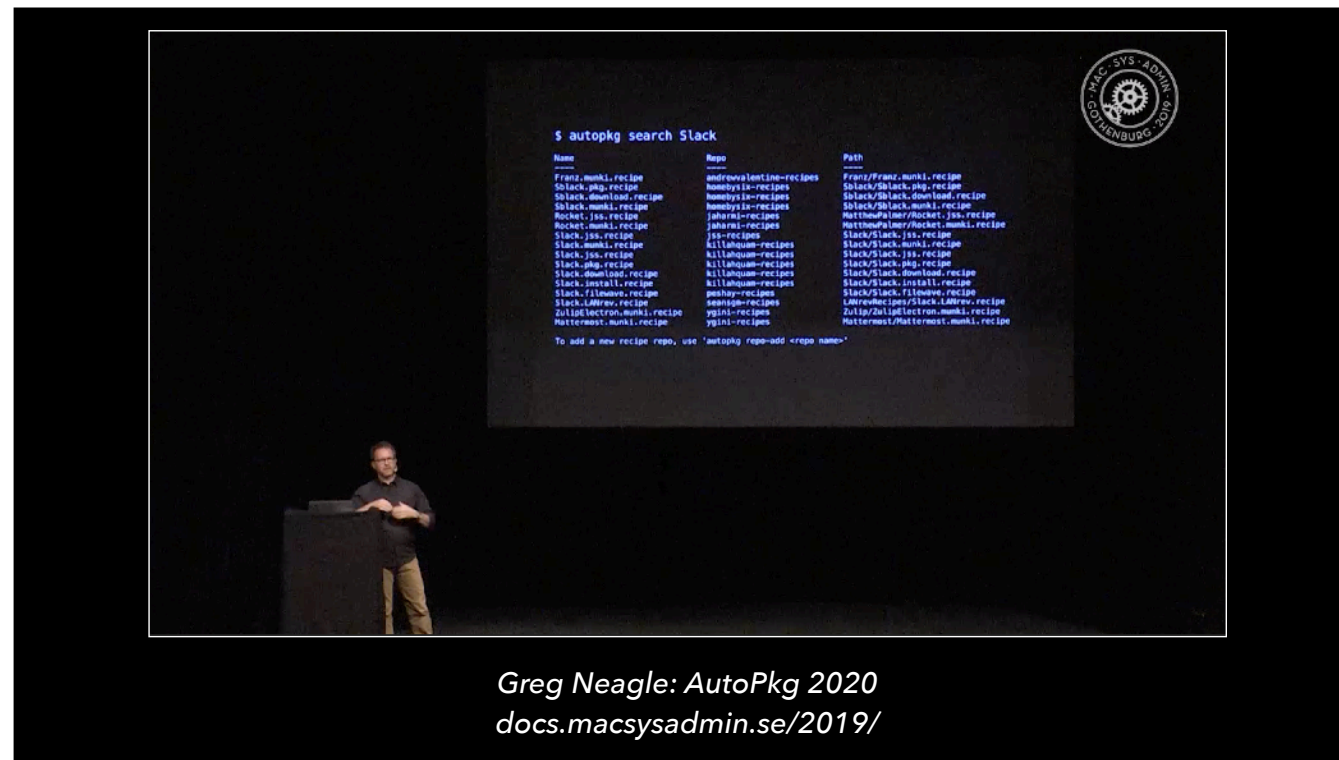
Why you should use AutoPkg

How to get started with AutoPkg

How to start writing recipes with AutoPkg



These are the things I am not discussing today, at least in any cohesive manner.



If you *do* want to learn more about AutoPkg, Greg Neagle's talk at last year's MacSysAdmin Conference is a fantastic starting point, particularly the first 20 minutes.

autopkg / autopkg

Watch108

Star762

Fork135

<> Code

Issues18

Pull requests2

Actions

Wiki

Security0

Insights

More resources

Anthony Reimer edited this page 3 hours ago · 14 revisions

Here are links to some additional resources surrounding AutoPkg and related tools:

Pages72

Blog posts

- Greg Neagle: [AutoPkg 2020: MacSysAdmin 2019 Links](#) - October 2019
- Rich Trouton's [posts](#) about AutoPkg on the Der Flounder blog - 2013–present
- Graham R Pugh's [blog](#) has posts about AutoPkg - 2015–present
- Anthony Reimer's Mac Labs [Blog](#) has posts about AutoPkg (marked with 🍷) - 2014–present
- Alan Siu's [posts](#) about AutoPkg on his blog - 2019–present
- Alan Siu's [posts](#) about AutoPkg on the St. Ignatius College Prep Tech Blog - 2015–2017
- Greg Neagle: [Using AutoPkg for "General Purpose" Packaging](#) - July 30, 2015
- Steve Wood's [posts](#) about AutoPkg on The Geeky Gordo blog - 2013–2015
- Steve Yuroff: [AutoPkg and Jenkins under one admin account](#) - Oct. 21, 2013
- Sean Kaiser: [AutoPkg change notifications](#) - Dec. 16, 2013

Table of Contents

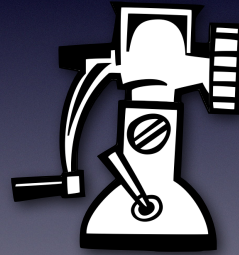
- [Introduction](#)
- [Getting Started](#)
- [FAQ](#)
- [More Resources](#)
- Notes: [MunkiImporter support for Munki repo plugins](#)
- AutoPkg Reference
 - [Preferences](#)
 - Recipes
 - [Recipe Format](#)
 - [Input Variables](#)

AutoPkg Wiki – More Resources Page

github.com/autopkg/autopkg/wiki/More-resources

There is also a lot of good information in the AutoPkg wiki, particularly *this* page that lists common blog posts and conference videos.

Creating Recipes



11

Writing and Understanding AutoPkg Recipes
github.com/jazzace/mactech-2019-autopkg

If you are specifically interested in recipe writing, I also did a talk at last year's MacTech Conference on the topic that was recorded live but the video has not been posted. I'm in the process of recording a narrated version of that presentation that I hope to make available sometime this month. It will be posted to the GitHub repo shown on screen alongside the other resources from the presentation that are already there.



Instead, the format of today's talk was inspired by a recent quick talk by Armin Briegel on 20 Obscure Terminal Features, which also became a blog post and a Twitter thread. So in the same way, this talk is a bowl full of nuggets that I've learned from the maintainers of AutoPkg. It's not as sharply focussed as Armin's talk, but I hope you'll find some gold in what I have to share today.

What I Am Discussing

Open Source Etiquette

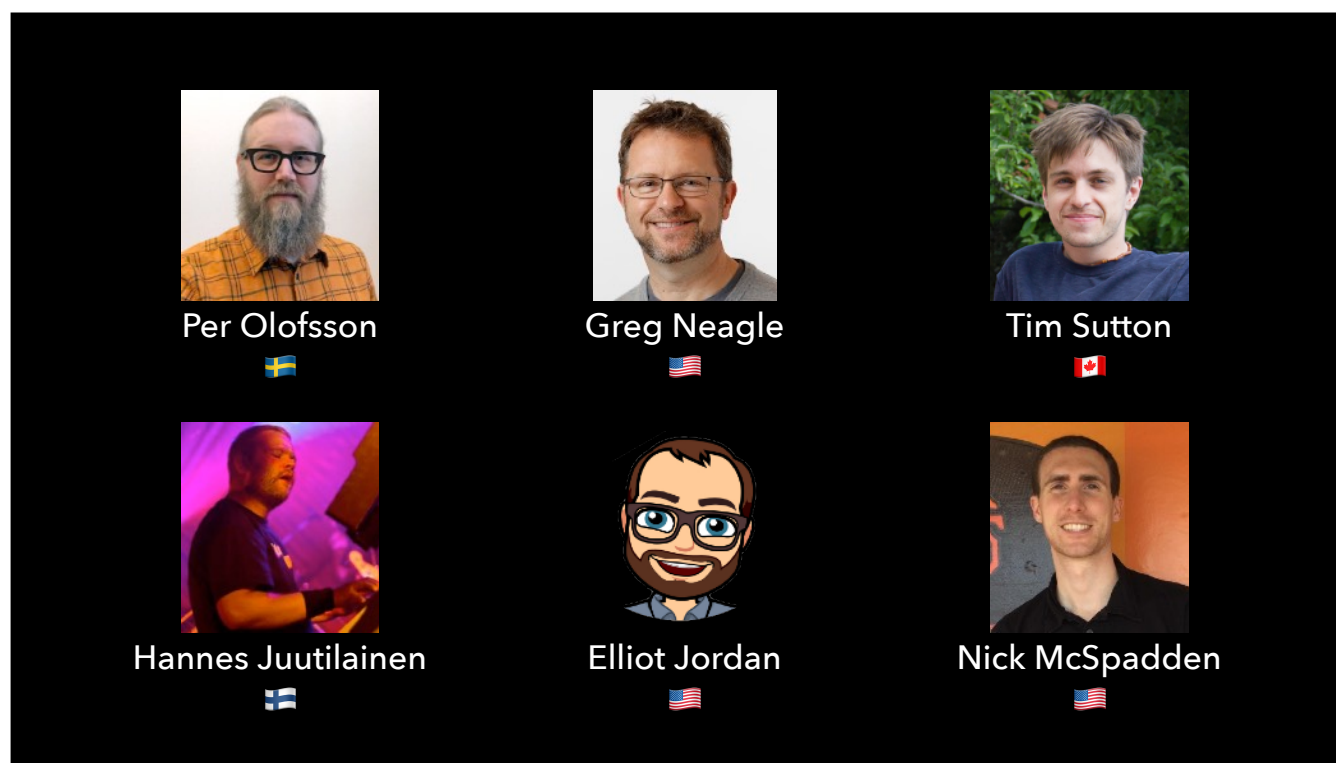
The Zen of AutoPkg

AutoPkg Recipe Writing Tips

Making AutoPkg More Secure



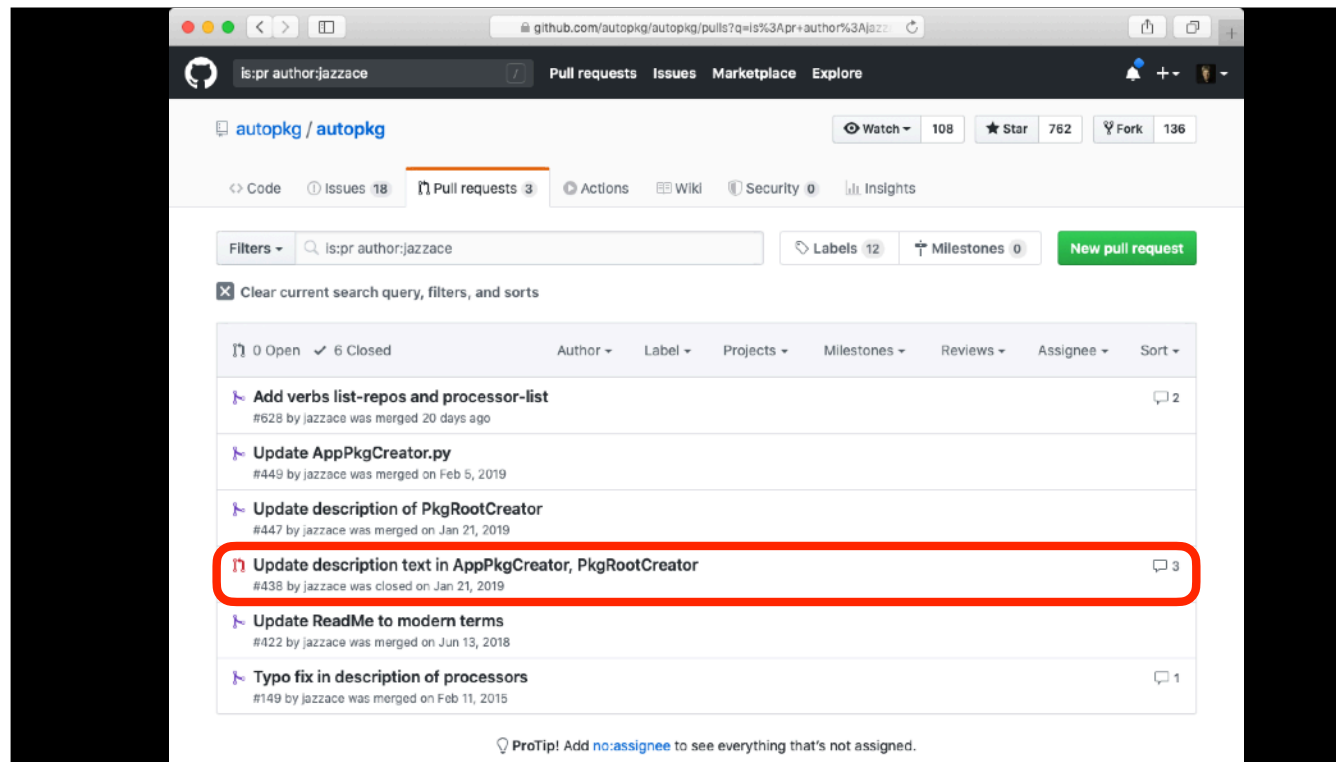
I've grouped the nuggets, if you will, into these four categories. Not all are specific to AutoPkg and some are more conceptual than technical. So even if you don't know AutoPkg, I'm hoping you'll still find something you can use.



Here are the people who inspired today's content. A big thank you to all of them for the work they do in the Open Source community, not just in AutoPkg.

Open Source Etiquette

So let's start with our first category. Once you start working with an open source product like AutoPkg in any sort of depth, you'll discover things that need to be fixed or enhanced. Often, they will be very small.



Here are the 5 pull requests I have contributed to the AutoPkg project, 2 of which edit prose and 2 fix minor typos. They are proof that anyone can give back to an Open Source project — updating documentation and taking care of details is a useful contribution. But there are 6 PRs in this list... 🍏 this one is closed. Here begins Learning Number 1:

Keep Your PRs Clean

[pause] I don't know about the rest of you, but I am not good at Git. Let me give you an example about how *not* to do a pull request. I decided to take the fork of AutoPkg I had created for a previous pull request and use it for a new one.

Update description text in AppPkgCreator, PkgRootCreator
#438

Closed

jazzace wants to merge 7 commits into autopkg:master from unknown repository

Conversation 3

Commits 7

Checks 0

Files changed 2

+3 -2

jazzace commented on Nov 17, 2018

Member

In the processor info for AppPkgCreator there was a space missing inside the quotes after "the" and before "CFBundleShortVersionString" in the description for the version input variable (Line 60).

Anthony Reimer and others added 7 commits on Jun 13, 2018

Merge remote-tracking branch 'autopkg/master'

fe25fa4

Documentation nomenclature update

b2f27e2

list-recipes fix wen piping encoded character

69c94d4

Revert "list-recipes fix wen piping encoded character"

723cc5f

Merge remote-tracking branch 'upstream/master'

033cec5

Update Code/autopkglib/AppPkgCreator.py

4b3a5f5

Reviewers

No reviews

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked issues

For a pull request that was intended to change literally one character, it had a lot going on... 🍏 It had remnants of my previous PR 🍏 It had some work on an obscure bug that I ended up reverting 🍏 It then had a merge with the upstream master, because my fork was woefully out of date before I applied the one-character fix. Not surprisingly, this request sat without comment for two months.

and before "CFBundleShortVersionString" in the description for the version input variable (Line 60).

Anthony Reimer and others added 7 commits on Jun 13, 2018

- Merge remote-tracking branch 'autopkg/master' fe25fa4
- Documentation nomenclature update b2f27e2
- list-recipes fix wen piping encoded character 69c94d4
- Revert "list-recipes fix wen piping encoded character" 723cc5f
- Merge remote-tracking branch 'upstream/master' 033cec5
- Update Code/autopkglib/AppPkgCreator.py 4b3a5f5
- Add to PkgRootCreator description. fa9dac9

jazzace commented on Jan 21, 2019

As per discussion on MacAdmins Slack on 2019-01-21, added text to the PkgRootCreator description to highlight that it can be used for generic directory creation.

jazzace changed the title ~~Add a space to AppPkgCreator~~ to **Add a space to AppPkgCreator-Update description text in AppPkgCreator, PkgRootCreator** on Jan 21, 2019

Assignees
No one assigned

Labels
None yet

Projects
None yet


Milestone
No milestone


Linked issues
Successfully merging this pull request may close these issues.
None yet

Notifications
Customize
[Unsubscribe](#)
You're receiving notifications because you modified the open/close state.


2 participants

Then, as an outgrowth of a discussion on the MacAdmins Slack, I added some text to a processor description. I thought, "I should just add it to my previous request." So I did, updating the title of the pull request to better describe the content of the request. And then, the teacher appeared...



 **jazzace**


 changed the title ~~Add a space to AppPkgCreator~~ Update description text in AppPkgCreator, PkgRootCreator on Jan 21, 2019



gregneagle commented on Jan 21, 2019

Member 😊 ...


Way too many changes/commits in this PR. The AppPkgCreator changes need to be squashed and a single commit (so much noise), and the PkgRootCreator change should be a different PR.




jazzace commented on Jan 21, 2019

Author Member 😊 ...

I'll start from scratch.



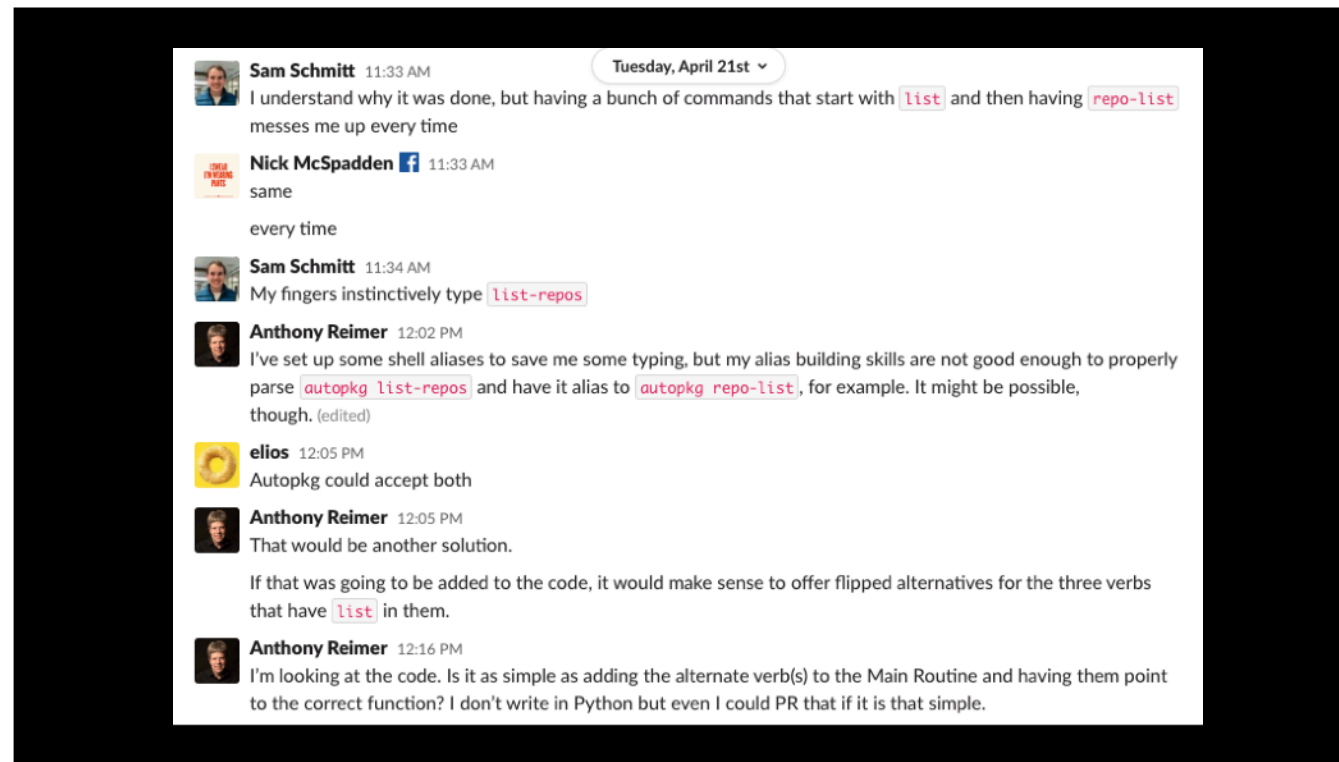
 **jazzace**

 closed this on Jan 21, 2019

[pause] Greg's guidance here is not only a good suggestion for Open Source projects, it's a good axiom for life. Less noise, more clarity. Keep separate items separate. Overall, just keep your PRs clean.

Show Your Work

Lesson number 2 is related to the rigour that Nick McSpadden brings to the table as one of the two most recent additions to the maintainer group. I've filed two PRs this year, one on the core recipes repo, one on the actual codebase. Nick looked at both of them. Here's the story behind the most recent one.



When a discussion in the MacAdmins Slack arose about how people commonly type the verb *list-repos* instead of the correct verb *repo-list*, it was suggested that AutoPkg could accept both. I looked at the code for a few minutes, saw that the fix appeared to be simple, and thought *even I* could PR this.

autopkg / autopkg

Watch 108Star

<> Code

! Issues 19

🔗 Pull requests 3

▶ Actions

📖 Wiki

🛡 Security 0

📊 Insights

Add verbs list-repos and processor-list #628


Merged nmcspadden merged 1 commit into `autopkg:dev_fetch_parents` from `jazzace:master` 21 days ago

💬 Conversation 2

🔗 Commits 1


🔍 Checks 0

📄 Files changed 1

 **jazzace** commented on Apr 21

Member 😊 ⋮

Alternative syntax for repo-list and list-processors

🔗  Add verbs list-repos and processor-list ⋮

0ee0753

Reviewer:

No review

Assignee:

No one as

Labels


So I did. But this pull request actually changed the function of AutoPkg, unlike the prose and typo fixes I had submitted earlier. So Nick responded.

Conversation 2

Commits 1

Checks 0



Files changed 1




jazzace commented on Apr 21

Member 😊 ...

Alternative syntax for repo-list and list-processors


  Add verbs list-repos and processor-list ... 0ee0753



nmcspadden commented on Apr 21

Member 😊 ...

At the very minimum, you will need to put more information into your PR. Please show some actual real-world tests that your command works, what the output looks like, etc.



jazzace commented on Apr 21 • edited ▼

Author Member 😊 ...

No problem. Running in zsh on macOS 10.15.4:

Reviewers

No review

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestones

No milestones

Linked issues

[pause] Even though this was a simple change, he wanted to see sample verbose code runs. This is what he requests of everyone, actually. I had run those tests anyway, so it was easy to do as asked.

autopkg / autopkg

Watch

108

Star

762

Fork

137

<> Code

Issues 19

Pull requests 3

Actions


Wiki

Security 0

Insights

Issue: Bug report

Report an issue with AutoPkg release. If this doesn't look right, [choose a different type](#).



Title

WritePreview

Describe the problem

A clear and concise description of what the problem is.

Preferences contents

BE SURE TO SANITIZE ANY SENSITIVE DATA SUCH AS PASSWORDS OR ADDRESSES.
Provide the output of `defaults read com.github.autopkg`, or the contents of your external `--prefs` file.

AutoPkg output

BE SURE TO SANITIZE ANY SENSITIVE DATA SUCH AS PASSWORDS OR ADDRESSES.
Provide the output of `autopkg run --vvvv <RecipeName>`, or any other command you are running. Please include as much data as possible.

Expected behavior

A clear and concise description of what you expected to happen. What specific part of the recipe or AutoPkg run did not behave correctly?

Version (please complete the following information):

- OS version: [e.g. 10.14.6, 10.15.1]
- AutoPkg Version: (generally expected that everyone should be on the latest release, but if you are using master or a specific commit, please specify)

Styling with Markdown is supported

Submit new issue

Helpful resources

[Contributing](#)

[GitHub Community Guidelines](#)

Remember, contributions to this repository should follow its [contributing guidelines](#).

Similarly, if I had been filing an issue with AutoPkg, I would have had a handy template to fill in that would have provided the details the maintainers wanted. This is a win-win proposition: if you provide the details the maintainers need, it makes their job easier, making it more likely that your issue will be addressed or...

Search or jump to... / Pull requests Issues Marketplace Explore

autopkg / autopkg Watch 108 Star 762 Fork 137

<> Code Issues 19 Pull requests 3 Actions Wiki Security 0 Insights

Releases Tags

Latest release

v2.1
54d9b78

Compare

AutoPkg 2.1

nmcspadden released this 9 days ago · 1 commit to master since this release

2.1 (May 19, 2020)

NEW FEATURES
AutoPkg now supports the verbs `list-repos` and `processor-list` for convenience (#628)

`autopkg info --pull / -p` now allows you to fetch all parent repos of a recipe automatically.

Example:

```
$ autopkg repo-delete recipes
$ autopkg info -p GoogleChrome.munki
Didn't find a recipe for com.github.autopkg.munki.google-chrome.
Found this recipe in repository: recipes
Attempting git clone...
Adding /Users/nmcspadden/Library/AutoPkg/RecipeRepos/com.github.autopkg.recipes to RECIPE S
```

...maybe, just maybe, you will have the enjoyment of seeing one of your contributions listed in the release notes.

Keep Your PRs Clean

Show Your Work

The overarching theme: make it as simple as possible for the maintainers to accept your request. Less work for them, a better product for everybody. And if you see something that could be better, put it on yourself to try to make it happen. The code was freely given to you. Give back when you can.

The Zen of AutoPkg

[8 min elapsed] The next set of learnings are conceptual in nature, more of a way to *think* about AutoPkg than a set of How-To tips.

Recipes

When we talk about recipes, we generally categorize them in a couple of ways...

Recipes

.download
.munki .pkg
.jss
.filewave
.install
Parent
Child
Override
Stub
XML
YAML

We describe them by **function**: what they produce or what they do with what is produced. These are the **nouns and verbs** of AutoPkg, if you will. 🍏 And we describe their **characteristics**: how they relate to each other and their technical function. 🍏 Soon, we may even be describing which markup language is inside. These are the **adjectives** of AutoPkg. While those concepts may be useful to help you talk about AutoPkg in a clearer, more literate way, that's not my learning from the maintainers. I go back to this thought from Greg Neagle...



[pause] This is an idea that's particularly applicable when we are talking in adjectives. A child recipe is not a special type of recipe, it's a recipe that just happens to have a parent. An override is a kind of child recipe that, when created, has no function other than to customize input variables and, more recently, hold trust information. But could you add processor steps to it, if you really wanted to? Absolutely, because it's just a recipe. A stub recipe is meant to point at something else, whether that's a shared processor or a parent recipe. But it can also be used to change the default value of an input key in the parent recipe, like the core AutoPkg recipes for Microsoft Office do, where you have a separate recipe for Excel and Word, for instance, that have no processor steps. If you understand that they're all just recipes, not only does it demystify some of the aspects of *how* AutoPkg works, it opens up new possibilities on how you can *make* it work.



Keep it Simple (for users)

Another one of the things you saw right from the beginning in AutoPkg was the use of custom processors in recipes to make the input variables simpler and more meaningful for potential users. If you're like me and can't code in Python (yet), you can do a heck of a lot with the core processors and the processors others have written,...

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Description</key>
  <string>Fetches the latest Sassafras K2 Mac installer specified by PRODUCT
(usually Admin or Server).

REVISION is the major version without a decimal, for example:
7.0: 70
7.1: 71
7.2: 72
7.5: 75

This recipe supports only a REVISION of 70 and up. If REVISION
is set to an empty string, the latest version will be retrieved –
this is generally the desired setting.

PRODUCT must be one of the following:
Admin
Server
Client

The primary purpose of this recipe is to support Admin and Server installer downloads.
If you want to customize your Client installer, you should use the SassafrasK2Client recipes.
(If, however, you want an unaltered client installer, like those required to support the
self-updates functionality, then this is the recipe for you!)

If you want to download more than one (e.g., both Admin and Server),
make an override for each and specify a unique identifier using the -n option
(e.g., autopkg make-override SassafrasK2.download -n SassafrasK2Server.download).
</string>
```

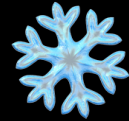
github.com/autopkg/jazzace-recipes/blob/master/Sassafras/SassafrasK2.download.recipe

...but that often means that you have to document the heck out of what input values are acceptable because you don't have the ability to use sane defaults or do error checking. Either way, you are not only trying to make this automation simpler for others, you're also making it simpler for future You. So hide the complexity if you can but document it when you can't.



Everything is created/used
in context

Even though you may observe that the AutoPkg community is all about sharing, one must also realize that people who create recipes for AutoPkg make them to serve a particular need that *they* have. Sometimes, that need is different than what *you* need. I'm thinking specifically about some of the post-processing one may need to do after a download. AutoPkg is great at automating that kind of stuff, but maybe you need more or less customization than the original recipe author. It's also a community principle that we should avoid duplication of recipes when possible, so what should you do when you need something different?



But I want something *different!*

- Child recipe
- Engage original author (directly or by PR)
- Build something new

First, you can build on the existing work by making a child recipe that either adds the additional steps you need, or replaces a different child in the recipe chain. If you saw my 2019 Writing Recipes talk, I described one such example in detail. If you create a new child recipe and share it, I encourage you to mention the *why* of what you did in the recipe description so that others can decide whether they want to use yours or the original. 🍎 Also, if you see a way that your desired functionality can be added without losing any ability of the original recipe, then talk to the author or even do the work and make a pull request. The only caveat here is that the other author is going to have to support the changed recipe, so reaching out can help determine whether your change might be welcomed. 🍎 But if all else fails, or if you need *less* or *different* functionality, sometimes you just need to build a new recipe for yourself. That's not ideal, but it's OK...

```
PRODUCT must be one of the following:  
Admin  
Server  
Client
```

The primary purpose of this recipe is to support Admin and Server installer downloads. If you want to customize your Client installer, you should use the Sassafrask2Client recipes. (If, however, you want an unaltered client installer, like those required to support the self-updates functionality, then this is the recipe for you!)

If you want to download more than one (e.g., both Admin and Server), make an override for each and specify a unique identifier using the -n option (e.g., autopkg make-override Sassafrask2.download -n Sassafrask2Server.download).

github.com/autopkg/jazzace-recipes/blob/master/Sassafras/Sassafrask2.download.recipe

As with the child recipe option, just document it so that others know why they would want to use your version over a different one. What you currently see on screen is an example of such documentation from the recipe description I showed a minute ago.

They're all just *recipes*
Keep it Simple (for users)
Everything is created/used in context

So if you try to understand the overall philosophy behind AutoPkg, you can be much more Zen when you encounter things that used to puzzle you. And remember as well, AutoPkg is a sharing community. Give back when you can, even if that is answering a question for a newbie in the MacAdmins Slack.

AutoPkg Recipe Writing Tips

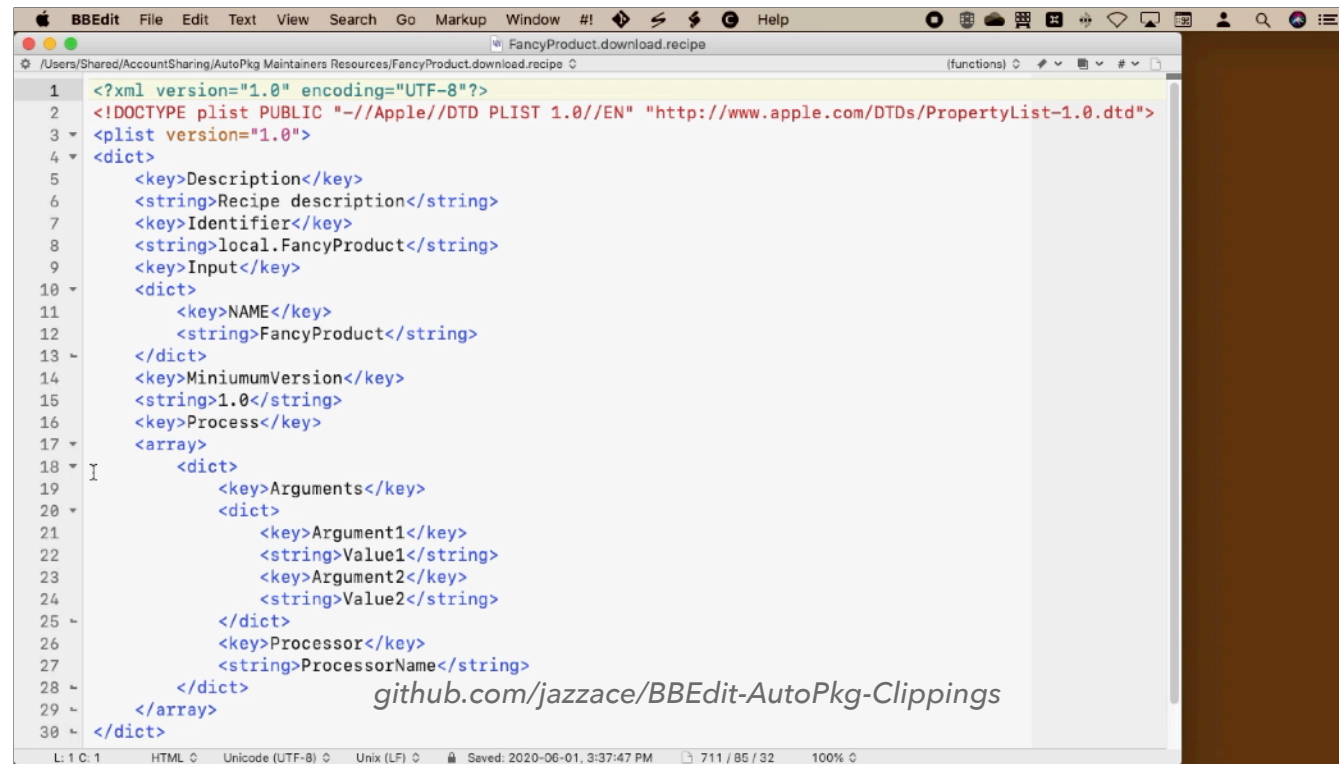
[13 min gone; 17 min left] Finally, for those of you waiting for the technique portion of this presentation, congratulations on your persistence. We have arrived!

Leverage your text editor

First, whichever plain text or code editor you use, there will be opportunities to automate or facilitate your recipe writing. Here's a couple of things I've done with my configuration of BBEdit to help, mimicking some of Elliot's practices.

```
<dict>
  <key>Processor</key>
  <string>URLTextSearcher</string>
  <key>Arguments</key>
  <dict>
    <key>url</key>
    <string>https://</string>
    <key>re_pattern</key>
    <string>(</string>
    <key>result_output_var_name</key>
    <string>url</string>
  </dict>
</dict>
```

One of the common processors I use is URLTextSearcher. I don't always remember the exact syntax of the arguments, so I generally look it up using the autopkg processor-info command or I refer to an existing recipe that uses it. BEdit has a feature called Text Clippings, where I can put any amount of boilerplate text in a file and ask for it to be inserted on demand. Here's a quick demo...

A screenshot of the BBEdit text editor on a Mac. The window title is 'FancyProduct.download.recipe'. The menu bar includes Apple logo, BBEdit, File, Edit, Text, View, Search, Go, Markup, Window, #!, and Help. The status bar at the bottom shows 'L: 1 C: 1', 'HTML', 'Unicode (UTF-8)', 'Unix (LF)', 'Saved: 2020-06-01, 3:37:47 PM', '711 / 85 / 32', and '100%'. The main text area contains an XML plist document. The code is as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
3 <plist version="1.0">
4   <dict>
5     <key>Description</key>
6     <string>Recipe description</string>
7     <key>Identifier</key>
8     <string>local.FancyProduct</string>
9     <key>Input</key>
10    <dict>
11      <key>NAME</key>
12      <string>FancyProduct</string>
13    </dict>
14    <key>MiniumumVersion</key>
15    <string>1.0</string>
16    <key>Process</key>
17    <array>
18      <dict>
19        <key>Arguments</key>
20        <dict>
21          <key>Argument1</key>
22          <string>Value1</string>
23          <key>Argument2</key>
24          <string>Value2</string>
25        </dict>
26        <key>Processor</key>
27        <string>ProcessorName</string>
28      </dict>
29    </array>
30  </dict>
```

[video paused] Here's the start of a new recipe. For the purposes of this demo, I'm going to use a recipe created by the new-recipe verb, but you're more likely going to copy an existing recipe. You'll notice that the new-recipe verb starts with a generic processor. That might be a useful thing to use while editing, so let's capture it as a clipping. 🍏 [video starts] I simply select it, use a contextual menu or go to the Clippings menu and select "Save Selection as Clipping...". I'll call it Generic Processor and put it in the set I have already created for AutoPkg. Now, I can place the insertion point at the beginning of the line and insert a clipping from the ones I've created. I can do it two ways: I can go to the Clippings menu and select it from the menu, or I can just start typing and BBEdit will offer to insert the clipping based on the clipping title. Pretty slick. 🍏 I'm posting my clippings to a public GitHub repo, so feel free to get a head start by grabbing mine and adding them to BBEdit or your favourite text editor.

Syntax and Standards

One of the things I really didn't consider much when writing recipes was adhering to standards. My recipes were legible, but since I borrowed from other recipes, sometimes indenting was done with spaces, sometimes with tabs, sometimes with both. It's better to have consistency for maintenance and support reasons, which is why there are now code standards if you are adding to the actual code base. You can apply standards with your recipes as well.

Syntax



```
plutil -lint /path/to/recipe
```



Markup > Check > Document Syntax

⌘Y

First of all, you'll want to check that you haven't made any syntax errors in your recipe. You can check it with this command in Terminal, but many pro text editors have syntax checkers for markup languages. 🍏 In the paid version of BBEdit, for example, the HTML syntax checker will work just fine for recipe purposes. That way, you can check your syntax without ever leaving the editor.

Standards



```
plutil -convert xml1 /path/to/recipe
```



Text > Apply Text Filter > *Filter Name*

```
plutil -convert xml1 - -o -
```

Then, you'll want to standardize the formatting of your recipe. Once again, the `plutil` command is your friend. It does a really nice job of cleaning up your recipe by standardizing indents with tabs and alphabetizing the dictionary keys at each level, but be aware that it will strip out any XML-style comments. Again, your text editor may be able to process the file you have open in the foreground. 🍏 BBEdit has Text Filters, where you can create a script in any number of languages, to act on the selected text, or the whole document if there is no selection. In this case, if you create a shell script with the command shown, you can apply this Text Filter as the last step in writing your recipe. Note that if there is a syntax error in your recipe, the conversion will fail, which makes the syntax check a good idea, especially since it's a single keystroke in the case of BBEdit.

Standards

```
<dict>
  <key>Processor</key>
  <string>DeprecationWarning</string>
  <key>Arguments</key>
  <dict>
    <key>warning_message</key>
    <string>This recipe is no longer supported.
      Expect it to be removed at a future date.
    </string>
  </dict>
</dict>
```

One of my pandemic projects was to clean up my own recipe repo in the project. I followed the steps I just enumerated for each recipe. My sincere apologies to all who were using my recipes who had to upgrade trust on all of the related overrides. 🍏 I also deprecated some recipes with the recent DeprecationWarning processor, something I recommend you do for all your recipes that are no longer relevant or have been superseded.

Standards

```
<dict>
  <key>Arguments</key>
  <dict>
    <key>re_pattern</key>
    <string>()</string>
    <key>result_output_var_name</key>
    <string>url</string>
    <key>url</key>
    <string>https://</string>
  </dict>
  <key>Processor</key>
  <string>URLTextSearcher</string>
</dict>
```

But one of the side effects of the clean up is that dictionary keys are alphabetized. To be honest, having the processor key at the end of the arguments is the closest I have ever come to understanding what people with dyslexia must go through. I struggle being able to read recipes this way. So my BBEdit Text Filter was a little different than what I showed previously...

Standards

OK, so I cheated

```
#!/bin/sh  
sed s!\>Processor\</!\>AAAAProcessor\</!g | \  
  plutil -convert xml1 - -o - | \  
sed s!\>AAAAProcessor\</!\>Processor\</!g
```

I used the substitution command *sed* to make Processor come before Arguments until the keys were sorted. To be clear, that shell command is entered as a single line in my script — I've added the green backslashes you see on screen to indicate this, but you don't type those in if you are entering it on a single line.

Disclaimer

From Elliot

I will now do my MacAdmins duty and present the argument Elliot made to me about the benefits of sticking with the simple conversion version over my Processor-first version:

1. Any recipes generated by Python's plistlib can be directly compared very easily since their dict key ordering and whitespace are consistent
2. Recipes not created programmatically can be brought into alignment with plutil -convert, no Python knowledge needed
3. Like Python black [the standard adopted for the AutoPkg codebase], the adherence to a standard (even an imperfect one) allows focusing on function over form

Leverage your text editor
Clippings • Syntax • Standards

So whatever you do, it would be beneficial if you did standardize your recipes in one of these ways. Leveraging the features of your text editor can make all of this better moving forward.

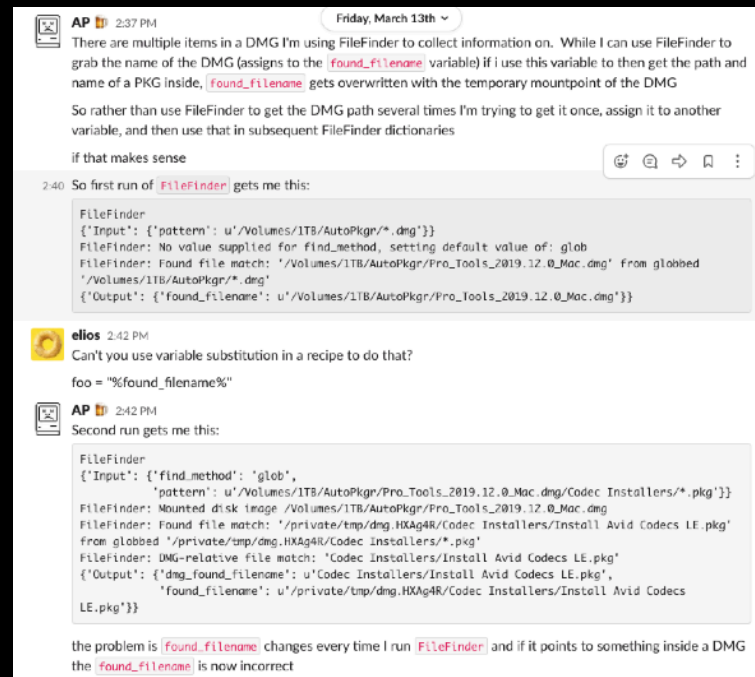


Variable names are arbitrary

[10 min left] This next tip is more useful than you might think. Remember how I mentioned earlier that cleaning up recipes with `plutil -convert xml1` would strip out XML-style comments? There is a workaround.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Comment</key>
  <string>Created with Recipe Robot v1.1.2 (https://github.com/homebysix/recipe-robot)</string>
  <key>Description</key>
  <string>Downloads the latest version of 8x8 - Virtual Office.</string>
  <key>Identifier</key>
  <string>com.github.arubdesu.download.8x8-VirtualOffice</string>
  <key>Input</key>
  <dict>
    <key>DOWNLOAD_URL</key>
    <string>https://support.8x8.com/@api/deki/files/</string>
    <key>NAME</key>
    <string>8x8 - Virtual Office</string>
  </dict>
  <key>MinimumVersion</key>
  <string>1.0.0</string>
<!-- This comment gets deleted if you run plutil -convert xml1 -->
```

If you create a key named Comment, whether as part of the top level dictionary (like Recipe Robot does) or when you are describing what you are accomplishing with a processor in the Process section of the recipe, it won't get stripped when you run your cleanup task. It also won't negatively affect the run; it will basically be ignored. In fact, even the word "Comment" is completely arbitrary — it could be anything not already in use. But it gets better...



Back in March, the admin known as APizz (because you don't want to hear me try to pronounce his full name) was writing a recipe that required running the FileFinder processor twice, but he needed to retain the output variable from each call for later use. After the second call of the processor, the first output variable value was overwritten. So he went to Slack to ask for assistance. In the ensuing discussion, one of the developers pointed out that variable names are arbitrary, otherwise Input variables wouldn't work....

```
<dict>
  <key>Comment</key>
  <string>Get the Pro Tools Codecs filename</string>
  <key>Arguments</key>
  <dict>
    <key>previous_found_filename</key>
    <string>%found_filename%</string>
    <key>pattern</key>
    <string>%found_filename%/Codec Installers/*.pkg</string>
  </dict>
  <key>Processor</key>
  <string>FileFinder</string>
</dict>
```

...so why not define a new variable after the first FileFinder call and assign it the value returned from that first call? That new variable can now be used later for string substitution. Many minds were blown that day.

autopkg / autopkg

Watch

108

Star

764

Fork

137

<> Code

🔔 Issues 20

🔗 Pull requests 3

🔄 Actions

📖 Wiki

🔒 Security 0

🔍 Insights

Defining Custom Variables

AP Orlebeke edited this page on Mar 26 · 3 revisions

EditNew Page

Defining & using custom variables in recipes

Pages 72

Overview

At the end of the day, autopkg input and output variables are just arbitrary strings that are used as a substitute for another string value.

While processors define a set of required and optional variables that can be used within its arguments, you can define your own custom variables within this structure. This is valuable when you need to use a particular processor multiple times within a recipe but need to persist the collected values, as on each subsequent processor run the previous output variable values get overwritten.

Typically, recipes do not need to persist the previous values of a given output variable. For example, a recipe needing to find multiple files (`FileFinder`) and copy them to a central location (`Copier`) in order to create a PKG, you can simply run the necessary processors for one item and repeat those processor steps for each additional item. Once you've completed the larger task of copying the files from the collected paths, storing what the previous values were for each item is unnecessary.

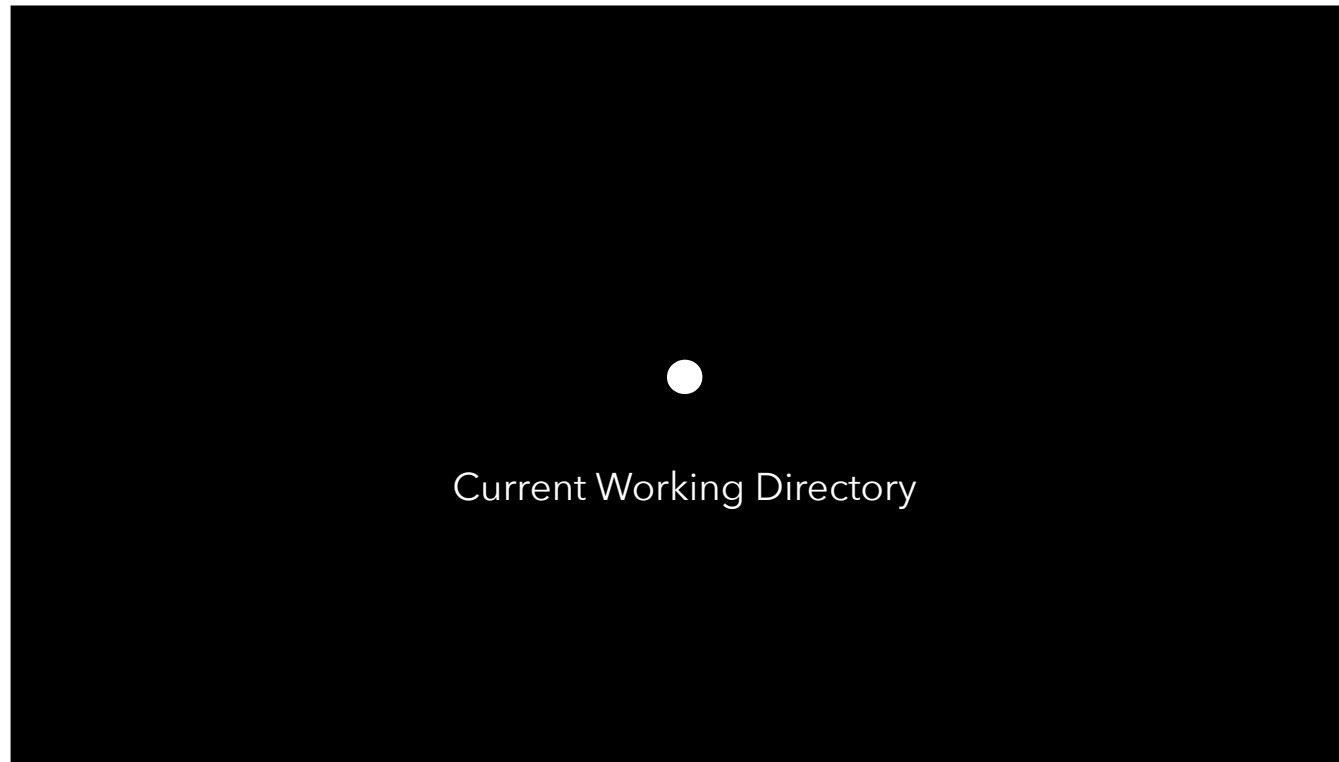
Defining custom variables

Defining a custom variable is as simple as supplying it as a key in the Arguments dictionary and setting it's value to the previously collected output variable.

Table of Contents

- [Introduction](#)
- [Getting Started](#)
- [FAQ](#)
- [More Resources](#)
- Notes: [Munkimporter support for Munki repo plugins](#)
- [AutoPkg Reference](#)
 - [Preferences](#)
 - [Recipes](#)
 - [Recipe Format](#)
 - [Input Variables](#)
 - [Important Variable Names](#)
 - [Recipe Search Order](#)
 - [Recipe Naming Conventions](#)
 - [Recipe Overrides](#)
 - [Recipe-writing Guidelines](#)
 - [Using CodeSignatureVerify](#)

And, of course, because APizz understands the Zen of AutoPkg, he gave back by documenting this in the Wiki so that everyone can learn about this, whether they were part of the discussion or not, whether they came to a talk like this or not.



This one I learned from Tim. [pause] Tim is very succinct. 🍏 In this case, the dot refers to the current working directory in your Terminal session. Here's why that's important and how you can leverage it, especially if you are testing a new or modified recipe.

```
% defaults read com.github.autopkg RECIPE_SEARCH_DIRS
(
  ".",
  "~/Library/AutoPkg/Recipes",
  "/Library/AutoPkg/Recipes",
  "/Users/areimer/Library/AutoPkg/RecipeRepos/
com.github.autopkg.recipes",
  "/Users/areimer/Library/AutoPkg/RecipeRepos/
com.github.autopkg.jazzace-recipes"
)
```

*More details in the Wiki:
github.com/autopkg/autopkg/wiki/Recipe-Search-Order*

AutoPkg has a defined search order when it is looking for recipes. It checks the override directory (or directories) for recipes first, then it uses the array of directories stored under the RECIPE_SEARCH_DIRS key in AutoPkg's preferences. When you add a repo using the repo-add verb or, if you are an AutoPkgr user when you *check* a repo in the top half of the Repos & Recipes tab, that repo is appended to the RECIPE_SEARCH_DIRS list. 🍏 But take a look at the first two directories in the list. A recipe in the current working directory will get read before everything except an override. This is important in AutoPkg because once AutoPkg finds a match for the recipe requested, it stops looking. So search order is important. In previous talks, I've spoken about the special properties of the user's Recipes folder but never gave the technical reason behind it. This is the reason. You can leverage this knowledge when writing a new recipe or rewriting an older one. My current practice is to place any recipe I'm editing into my user Recipes folder and set aside any override so that AutoPkg finds the version I'm editing first, but I could just as easily use any other directory as long as I make it my working directory using the cd command. While this feature dates back to the days before AutoPkg had formal repos, you can still leverage it today.

Leverage your text editor
Syntax and Standards
Variable names are arbitrary
Recipe Search Order (.)

There are lots of technical things I've learned from the people who wrote and maintain the code. I hope you can apply some of these in your work with AutoPkg.

Making AutoPkg More Secure

[6 minutes left] I want to finish up by quickly acknowledging Hannes and Elliot for not only raising awareness of possible issues with AutoPkg, but doing something about it.

"HTTPS downloads and verifying the signature are the most important steps a recipe author can do to enhance recipe security."

Hannes Juutilainen – July 2016 on autopkg-discuss

Hannes' largest contribution was the addition of the CodeSignatureVerifier processor in 2014. Whether a download might be compromised at the source or in transit, this check added a level of certainty that your recipes were fetching what was expected.




Common Download Recipe Workflow

1. Determine the URL of the item we want to download
2. Download a copy of the item
3. EndOfCheckPhase
4. Check the code signature of the downloaded item

Today, if you are writing a download recipe, it is expected that you will check for a code signature if one is present. Along the same lines, Hannes also has a VirusTotalAnalyzer post-processor in a personal GitHub repo. That uses the VirusTotal service to check for malware via an API.

MacDevOps June 20-21, 2016

#7: TRUST *without* VERIFICATION



- strong GitHub password?
two factor authentication for GitHub?
- strong email password?
two factor authentication for email?
- highly targeted or visible organization?
- sensible commit messages?
- strong Mac password?
- frequent repos changes?

How (Not) To Do Bad Things With AutoPkg – Elliot Jordan
www.youtube.com/watch?v=Q_cvgGtJ71M

Elliot Jordan came to MacDevOps Vancouver in 2016 to give what ended up being a landmark talk on 15 different bad things you could do with AutoPkg. I recommend viewing the 2017 MacAdmins Conference version of the talk, because by that point,...

Trust Verification

...AutoPkg had implemented Trust Verification, which really helped address some of the things on that list in a systemic way. Greg Neagle also talks about security features in the second half of the talk I cited off the top.

CodeSignatureVerifier

Trust Verification


With these two features, you can make your AutoPkg implementation far more secure.

	July '14	June '16	June '18	May '20
Total Recipes	630	4079	5936	8156
.download	258	1156	1667	2252
.munki	267	1037	1537	2125
.pkg	95	794	1205	1734
.install	0	596	723	892
.jss	8	269	430	607
.filewave	0	14	171	269


AutoPkg continues to thrive with more recipes being added every year. It is thanks to the developers and maintainers of the AutoPkg project we have such a mature and valuable automation platform to work with.


maclabs.jazzace.ca/2020/06/04/
things-i-learned-autopkg

Anthony Reimer

 maclabs.jazzace.ca

 [AnthonyReimer](https://twitter.com/AnthonyReimer)

 [jazzace](#)

 [jazzace](#)



UNIVERSITY OF
CALGARY