Solving problems with custom Autopkg processors

Matthew Warren (@haircut)

About me

Matthew Warren

- Systems Engineer at **Lyft**, primarily managing our global fleet of Macs.
- Writes **MacBlog.org**
- GitHub.com/haircut
- *haircut* on #macadmins Slack

🙋 Assumed Audience

You know how to use AutoPkg, and understand the basics of how recipes work.

Processor

AutoPkg tasks that make up a recipe

Processor == Python class

Built-in processors

Processor	Task
URLTextSearcher	"*find the text that looks like a download URL on this page*"
URLDownloader	"*download that file*"
Versioner	"*read the CFBundleShortVersionString attribute from the Info.plist file within the downloaded app*"

Input  …downloads/CatPicz.dmg/CatPicz.app

Action  AppPkgCreator

Output  CatPicz-1.2.49.pkg

Output

Variable representing the results of a processor’s actions

Built-in processor output

Processor	Variable	Example
URLTextSearcher	*%match%*	*https://example.net/downloads/latest/app.dmg*
URLDownloader	*%pathname%*	*/Users/haircut/Library/AutoPkg/Cache/com.github.autopkg.example/downloads/app.dmg*
Versioner	*%version%*	2.3.1

Processor context

Processors have access to:

1. User- or recipe-defined input variables
2. Output from previous processors

Everything is global.

A variable either exists or doesn’t.

…but the order of operations matters.

Shared Processors

Custom processors, shared like recipes

autopkg repo-add <repository>

Add the processor author's repository to your repo list.

Shared Processor Syntax

```
Process:
  - Processor: <stub recipe>/<processor>
```

Example

```
Process:
  - Processor: com.github.haircut.processors/DatetimeOutputter
```

Problem:

AutoPkg is *aware of* the date and time a recipes runs.

But that information is *not accessible to* a recipe itself.

DatetimeOutputter

Output the current date and time

…and optionally future or past dates and times.

DatetimeOutputter

Basic use

Process:
  - Processor: com.github.haircut.processors/DatetimeOutputter

```
“datetime”: “2022-05-18T13:45:45.544391”
```

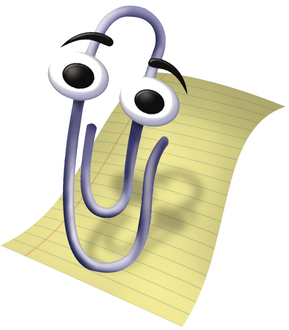
DatetimeOutputter

Datetime format

```
Process:
  - Processor: com.github.haircut.processors/DatetimeOutputter
    Arguments:
      datetime_format: "%A, %B %e, %Y at %l:%M %p”
```

```
“datetime”: “Wednesday, May 18, 2022 at 1:45 PM”
```

Try https://strftime.org
for date format reference



DatetimeOutputter

Basic use case

```
Process:
  - Processor: com.github.haircut.processors/DatetimeOutputter
    Arguments:
      datetime_format: "%A, %B %e, %Y at %I:%M %p”

  - Processor: com.github.grahampugh.jamf-upload.processors/JamfPackageUploader
    Arguments:
      pkg_notes: "Uploaded via AutoPkg %datetime%"
```

DatetimeOutputter

Basic use case

```
Process:
  - Processor: com.github.haircut.processors/DatetimeOutputter
    Arguments:
      datetime_format: "%A, %B %e, %Y at %I:%M %p”

  - Processor: com.github.grahampugh.jamf-upload.processors/JamfPackageUploader
    Arguments:
      pkg_notes: "Uploaded via AutoPkg %datetime%"
```

← CatPicz-1.2.49.pkg

General Options Limitations

Display Name Display name for the package

CatPicz-1.2.49.pkg

Notes Notes to display about the package (e.g. who built it and when it was built)

Uploaded via AutoPkg Thursday, May 18, 2022 at 1:45 PM

Time Deltas

Calculate past or future dates and times

DatetimeOutputter

Time deltas

```
- Processor: com.github.haircut.processors/DatetimeOutputter
  Arguments:
    format: "%Y-%m-%dT%H:%M:%SZ"
    deltas:
      - output_name: example_delta
        direction: future
        interval:
          weeks: 2
```

DatetimeOutputter

Munki: Force Install After Date

```
- Processor: com.github.haircut.processors/DatetimeOutputter
  Arguments:
    format: "%Y-%m-%dT%H:%M:%SZ"
    deltas:
      - output_name: force_after
        direction: future
        interval:
          weeks: 2

- Processor: MunkiImporter
  Arguments:
    MUNKI_REPO: <path to Munki repo>
    pkg_path: "%pkg%"
    pkginfo:
      force_install_after_date: "%force_after%"
```

DatetimeOutputter

Munki: Force Install After Date

```
- Processor: com.github.haircut.processors/DatetimeOutputter
  Arguments:
    format: "%Y-%m-%dT%H:%M:%SZ"
    deltas:
      - output_name: force_after
        direction: future
        interval:
          weeks: 2

- Processor: MunkiImporter
  Arguments:
    MUNKI_REPO: <path to Munki repo>
    pkg_path: "%pkg%"
    pkginfo:
      force_install_after_date: "%force_after%"
```

DatetimeOutputter

Jamf Pro: Policy Activation Dates

Recipe

```
- Processor: com.github.haircut.processors/
DatetimeOutputter
  Arguments:
    deltas:
      - output_name: activation_date
        direction: future
        datetime_format: "%Y-%m-%d 00:00:00"
        interval:
          weeks: 1

- Processor: com.github.grahampugh.jamf-
upload.processors/JamfPolicyUploader
  Arguments:
    policy_name: "Install %NAME% 1 Week After
Release"
    policy_template: "Install-after-1-week.xml"
```

Policy Template

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
    <general>
        <name>Install %NAME% 1 Week After
Release</name>
        <enabled>true</enabled>
        <trigger>CHECKIN</trigger>
        <trigger_checkin>true</trigger_checkin>
        <frequency>Once per computer</frequency>
        <date_time_limitations>
            <activation_date>%activation_date%</
activation_date>
        </date_time_limitations>
    </general>
...
```

DatetimeOutputter

Jamf Pro: Policy Activation Dates

Recipe

```
- Processor: com.github.haircut.processors/
DatetimeOutputter
  Arguments:
    deltas:
      - output_name: activation_date
        direction: future
        datetime_format: "%Y-%m-%d 00:00:00"
        interval:
          weeks: 1

- Processor: com.github.grahampugh.jamf-
upload.processors/JamfPolicyUploader
  Arguments:
    policy_name: "Install %NAME% 1 Week After
Release"
    policy_template: "Install-after-1-week.xml"
```

Policy Template

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
    <general>
        <name>Install %NAME% 1 Week After
Release</name>
        <enabled>true</enabled>
        <trigger>CHECKIN</trigger>
        <trigger_checkin>true</trigger_checkin>
        <frequency>Once per computer</frequency>
        <date_time_limitations>
            <activation_date>%activation_date%</
activation_date>
        </date_time_limitations>
    </general>
...
```

Problem:

I want to *grab the current icon from an app* to use in Jamf Pro Self Service.

But *I don’t want to manually extract it*, or worry about keeping it updated if the vendor changes it.

AppIconExtractor

Extract an app's icon

…and optionally create custom compositions.

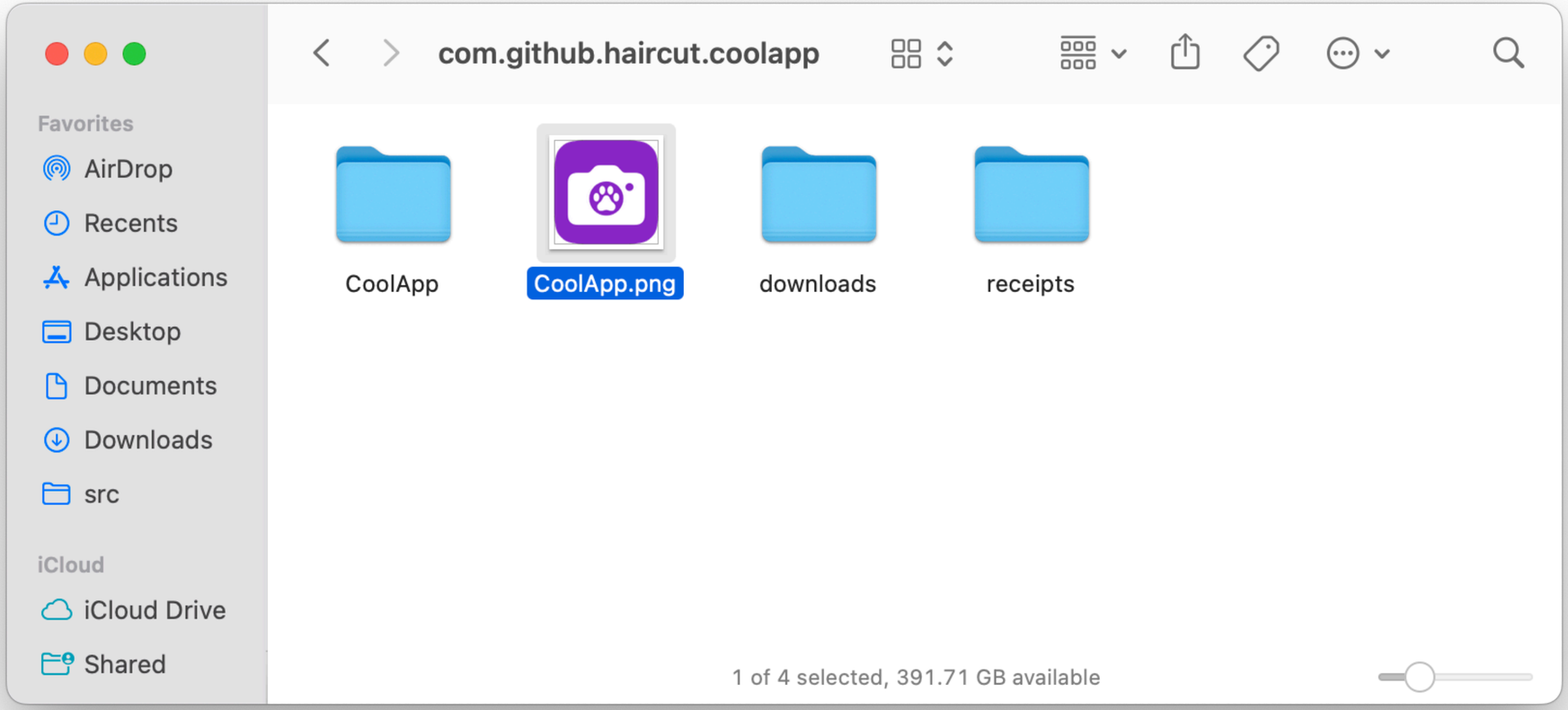
```
/usr/local/autopkg/python -m pip install --upgrade Pillow
```

AppIconExtractor

Basic use

```
Process:
  - Processor: com.github.haircut.processors/AppIconExtractor
    Arguments:
      source_app: "%RECIPE_CACHE_DIR%/downloads/CoolApp.dmg/CoolApp.app"
```

Extracted Icon



Output Variable

“app_icon_path”: “/Users/haircut/Library/AutoPkg/Cache/com.github.haircut.coolapp/CoolApp.png”

AppIconExtractor

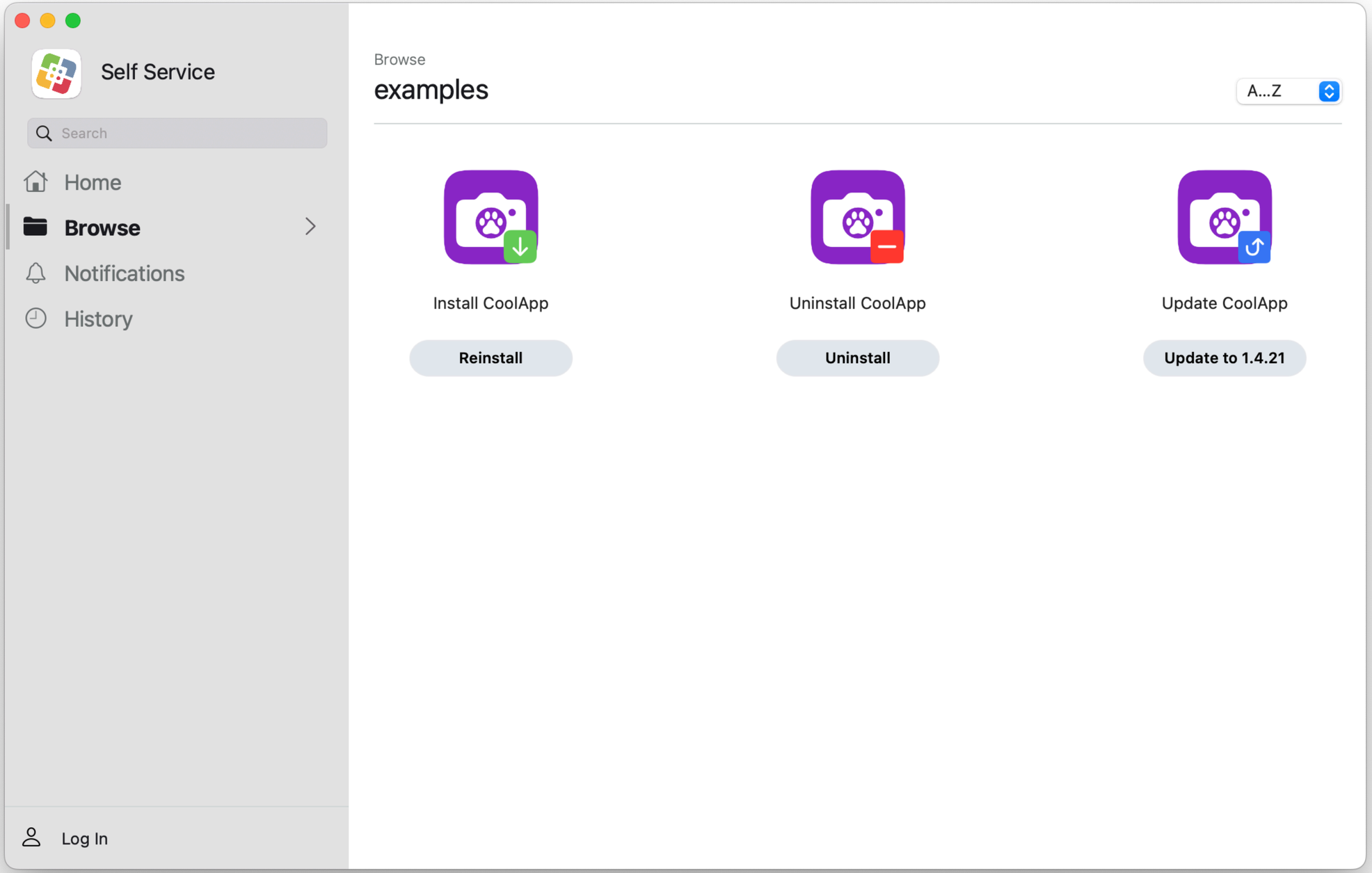
Uploading the icon to Jamf Pro

```
Process:
  - Processor: com.github.haircut.processors/AppIconExtractor
    Arguments:
      source_app: "%RECIPE_CACHE_DIR%/%NAME%/Sublime Text.app"

  - Processor: com.github.grahampugh.jamf-upload.processors/JamfPolicyUploader
    Arguments:
      policy_template: "%POLICY_TEMPLATE%"
      policy_name: "%POLICY_NAME%"
      icon: "%app_icon_path%"
      replace_icon: True
```

AppIconExtractor

Composite icons


Self Service
Search
Home
Browse
Notifications
History
Log In
Browse
examples
A...Z
Install CoolApp
Reinstall
Uninstall CoolApp
Uninstall
Update CoolApp
Update to 1.4.21

AppIconExtractor

Composite icons

```
Process:
  - Processor: com.github.haircut.processors/AppIconExtractor
    Arguments:
      source_app: "%RECIPE_CACHE_DIR%/CoolApp/CoolApp.app"
      icon_output_path: "%RECIPE_CACHE_DIR%/Icons/Icon-%NAME%.png"
      composite_install_path: "%RECIPE_CACHE_DIR%/Icons/Install-%NAME%.png"
      composite_update_path: "%RECIPE_CACHE_DIR%/Icons/Update-%NAME%.png"
      composite_uninstall_path: "%RECIPE_CACHE_DIR%/Icons/Uninstall-%NAME%.png"
```

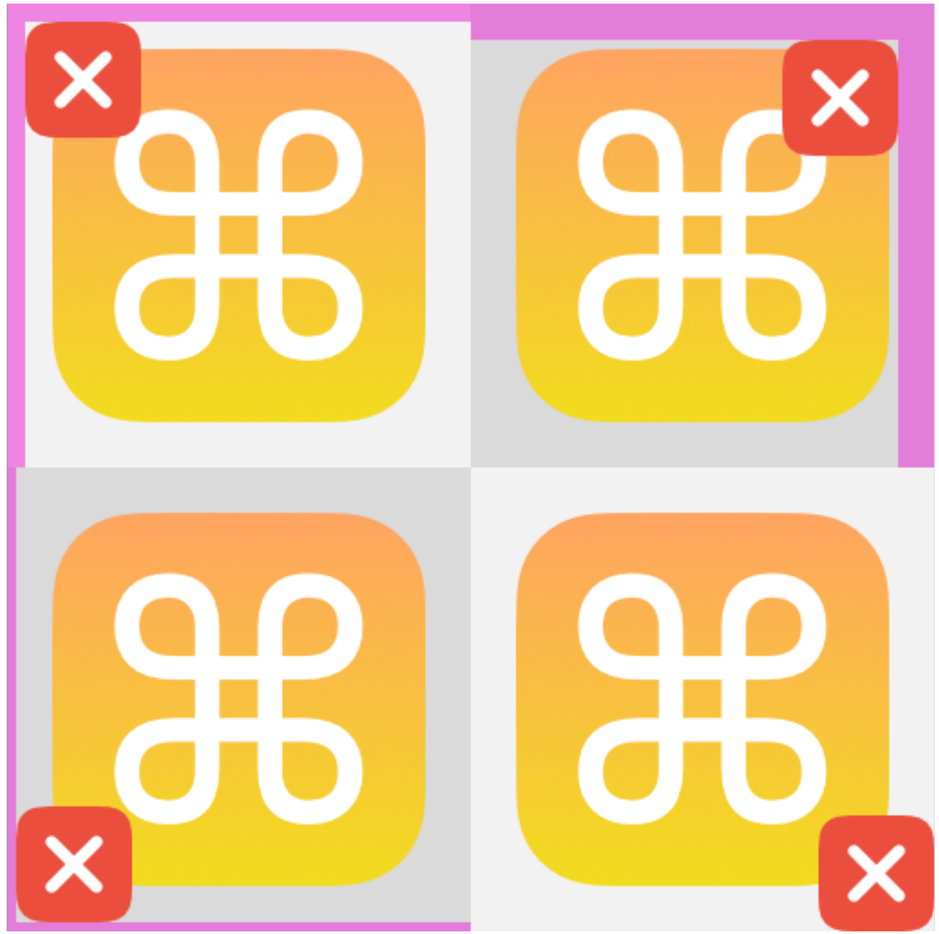
AppIconExtractor

Composite icons

```
Process:
  - Processor: com.github.haircut.processors/AppIconExtractor
    Arguments:
      source_app: "%RECIPE_CACHE_DIR%/CoolApp/CoolApp.app"
      icon_output_path: "%RECIPE_CACHE_DIR%/Icons/Icon-%NAME%.png"
      composite_install_path: “%RECIPE_CACHE_DIR%/Icons/Install-%NAME%.png"
      composite_install_template: “/Users/Shared/gift.png”
      composite_update_path: “%RECIPE_CACHE_DIR%/Icons/Update-%NAME%.png”
      composite_update_template: “/Users/Shared/hammer.png”
      composite_uninstall_path: “%RECIPE_CACHE_DIR%/Icons/Uninstall-%NAME%.png"
      composite_uninstall_template: “/Users/Shared/exit.png”
```

AppIconExtractor

Customizing composite icons



- composite_padding: an integer number of pixels of padding
- composite_position: br (bottom-right), bl (bottom left), ur (upper right), or ul (upper left)

Resources

macblog.org/datetime

macblog.org/icons

Resources

github.com/autopkg/autopkg/wiki/Noteworthy-Processors

Questions