jctl and python-jamf

James Reynolds, January 2023

Agenda

- The problem with Jamf
- The Jamf API
- Goals and history of the python-jamf
- python-jamf, jctl, pkgctl

What is Jamf?

- MySQL database
- Tomcat Server
- Java WAR app
- Angular front end (single page app)
    - The front end is the biggest problem
    - Lots of scrolling and clicking

WAR

ROOT.war

What Does Jamf Do?

- Package assignment and distribution
    - Jamf features that can install packages
        - Jamf Policies
        - Jamf Patch Management
        - Jamf Prestage Enrollment
- Demo

Jamf Policies

- Installs packages to “scoped” computers
- By design Jamf policies only run “once per computer”
  - Erratic distribution because of common errors
- Workaround
  - Run policies “ongoing”
  - Scope the policy to a smart group
  - This complicate everything and slows the server
- Demo

Jamf Patch Management

- Makes distributing packages really easy
- This wont install an application
    - Will only upgrade it if it’s already installed
    - Works great with Self Service otherwise…
    - Jamf policies are still required!
    - Just increases the number of steps to “promote” a package!
- Demo

Staging

- You have to have a smart group, a regular policy, and a patch policy for each application
    - 1 smart group, 1 policy, 1 patch policy
- Multiple stages (dev, testing, and prod) multiply the number above
    - 2 stages = 2 smart groups, 2 policies, 2 patch policies
    - A total of 6 things to edit for each package
- Demo

Solution: Jamf Classic API

- The Java Webapp powers the API
- There are 2 API’s and they don’t overlap very much
  - Jamf Pro API
  - Classic API (“classic” means it was their first attempt at an API)
    - Most of what we want to do is in the Classic API
    - Jamf is not going to develop the Classic API more
    - There’s problems with the JSON output?

What is OpenAPI/Swagger

- https://www.openapis.org/
- OpenAPI is a way to describe how to interact with an API
  - The definition is a JSON file
  - Create server and client code just from a Swagger file
  - Unfortunately, the Jamf Classic API doesn't actually conform to the standard so we couldn't get the auto-generated client to work (or maybe we just didn't know what we were doing)

Jamf’s Classic API Swagger File

33,691 lines

```
“paths": {
  "/accounts": {
    "get": {
      "summary": "Finds all accounts",
      "parameters": [],
      "produces": [
        "application/xml",
        "application/json"
      ],
      "responses": {
        "200": {
        "description": “OK",
```

curl -X GET "http://example.com/JSSResource/accounts" -H "accept: application/xml"

Jamf’s Classic API Swagger File

33,691 lines

```
"definitions": {
   "account": {
      "type": "object",
      "properties": {
         "id": {
            "type": "integer",
            "example": 1
         },
         "name": {
            "type": "string",
            "example": "John Smith",
            "description": "Name of the account"
         },
```

Jamf’s Classic API Swagger File

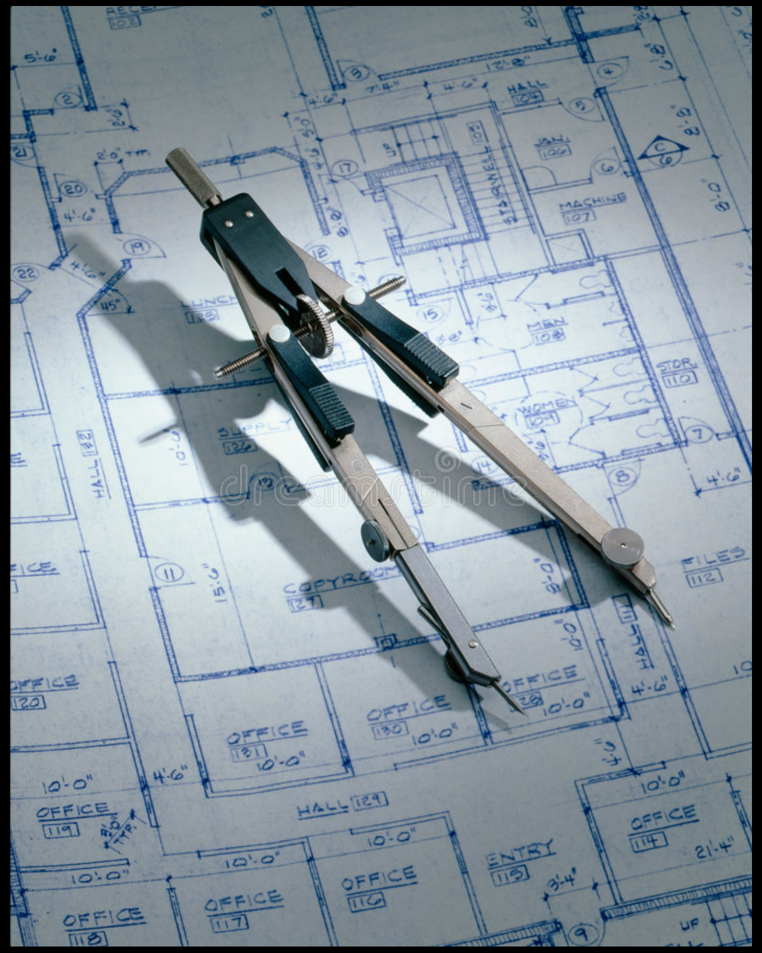
The result

```
<?xml version="1.0" encoding="UTF-8"?>
  <accounts>
    <users>
      <user>
        <id>2</id>
        <name>james</name>
      </user>
      <user>
        <id>1</id>
        <name>root</name>
      </user>
    </users>
    <groups/>
  </accounts>
```

Using the API

- It seems like a lot of people are using curl with bash scripts
- There are several libraries that interface with the API
    - RubyJSS
    - PythonJSS
- We don't really use Ruby and PythonJSS isn't maintained (and it had a high learning curve)
- So we reinvented the wheel

Design Goals



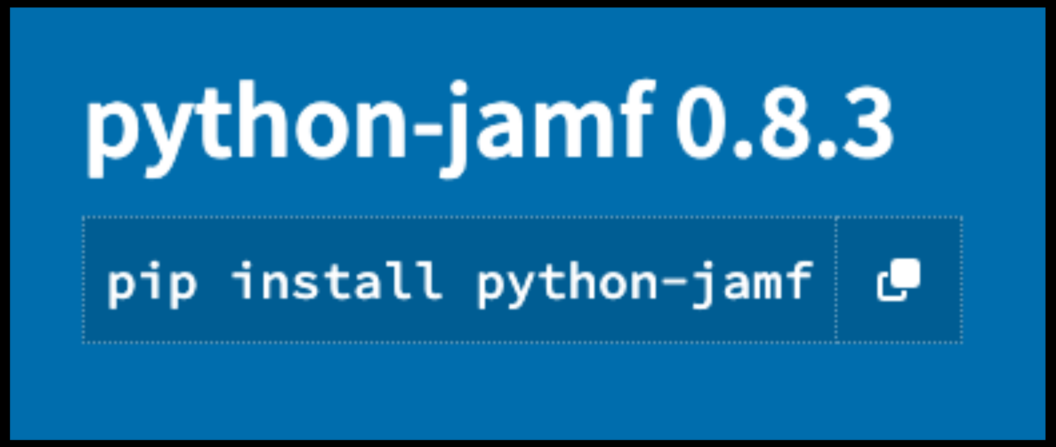
- Easy to use and understand
- Minimal maintainance
- Limit the scope
- Take shortcuts and do things “wrong” if it helps meet our goals
  - This mainly means that we aren’t creating Python objects for everything, it’s all just Python dictionaries (PythonJSS created objects for everything)

History

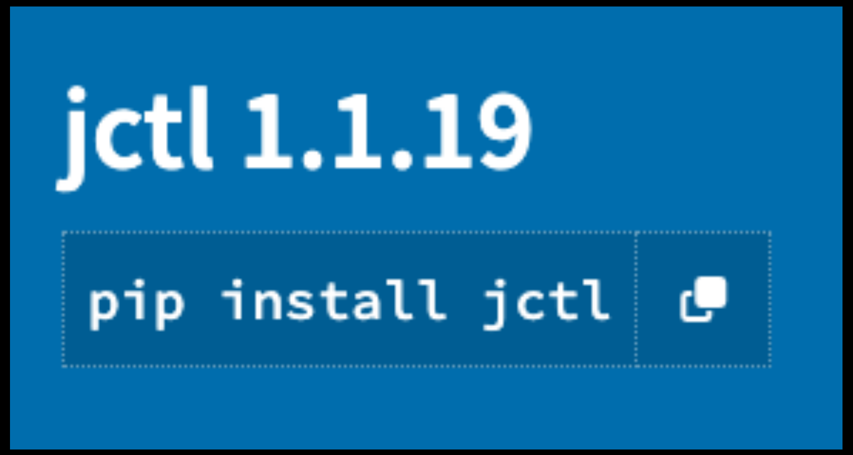
- Sam Forester wrote the foundation code before he took another job
- I helped Sam with the goals so I was already familiar with the project
- But I didn’t understand the Jamf API at all
- It was hardcoded for the Marriott Library’s needs
- I kept most of the code but also started over
- Tony Williams (github.com/honestpuck) gave us key contributions at the start to get us going in the right direction

What Have We Made?

- jamf/api.py
- jamf/records.py
- jctl
  - CLI CRUD (writing this taught me the Classic API)
- pkgctl
  - CLI package promotion
- Note: Runs on Linux and macOS


python-jamf 0.8.3

```
pip install python-jamf
```


jctl 1.1.19

```
pip install jctl
```

jamf/api.py

- HTTP get/post/update/delete methods
- Convert between XML and Python dictionaries
- Requires knowledge of the Jamf API
- Bearer token support
- Stores password in keyring (e.g. macOS keychain)

jamf/records.py

- Generic records classes with easy and common CRUD methods
- Much less knowledge needed
- Supported record types is limited

pkgctl
CLI package promotion

- Interactive promotion
- DEMO

jctl
CLI CRUD

```
jctl

jctl policies

jctl policies -r "Dropbox.*ing.*Test"

jctl policies -r "Dropbox.*ing.*Test" -l

jctl policies -r "Dropbox.*ing.*Test" -p general/category

jctl policies -u general/category=Hi
```

jctl

CLI CRUD

```
jctl categories

jctl categories -d

jctl categories -c Hi

jctl policies -u general/category=Hi

jctl policies -s general/category==Hi

jctl policies -s general/category==Hi -d
```

api.py

```
python3

import jamf

api = jamf.API()

cats = api.get(‘categories’)

cat = cats[‘categories']['category'][0]

cat = api.get('categories/id/1')

cat['name'] = 'Bye'

api.put('categories/id/1', {'category': cat})

api.post('categories/id/0', {'category': {'id': '0', 'name': 'Bye'}})
```

records.py

```
python3

import jamf

jamf.Categories()

cat = jamf.Categories().recordsWithName(“Hi")[0]

cat.data["name"] = “Bye"

cat.save()
```

Resources

- https://example.com:8443/api
- https://github.com/univ-of-utah-marriott-library-apple/python-jamf
- https://github.com/univ-of-utah-marriott-library-apple/python-jamf/wiki
- https://pypi.org/project/python-jamf/
- https://github.com/univ-of-utah-marriott-library-apple/jctl
- https://github.com/univ-of-utah-marriott-library-apple/jctl/wiki
- https://pypi.org/project/jctl/

Questions?