



CZI Grant Status

Zach Pearson
RBVI General Meeting
31 March 2022

Aim 1: Backwards Compatible Session Files

Right now this project is on the backburner as we prepare for the 1.4 release: Qt6 and webservices are the main focus right now. I'm gearing up for more effort on this project, though: previously its status was tracked in a Python file in the Chimera repo. I opened a ticket, #6505, to track it on Trac, and pasted the list of TODOs appearing in that file into the ticket discussion. It's optimistically milestone for 1.5. My plan is to convert the issues listed there to their own tickets and chip away at them one by one.

I've never used Chimera, and I wasn't around while that code was being written, so I have to thank Eric for being patient with me and explaining the challenges associated with getting this done. We'll work together to turn the TODOs into tickets and reevaluate at a later date.

Aim 2: Webservices REST Port



Status

- Modeller webservice has been ported
 - Status messages working well
 - Buffering never solved
 - Switched to using file counting to avoid `subprocess.run(..., shell=True)`
- Webservices and webservices-test now run in their own virtual environments (thanks, Greg!)
- Logs now written to correct directories: `$(LOGS)/webservices{,-test}/cxservices/`
- Most output files now readable by group 'ferrin' for debugging
 - One last issue: `os.mkdir` does not respect `setegid`, so Modeller directories are 'apache:apache' not 'apache:ferrin'
- Still needed:
 - Infrastructure documented on RBVI Trac
 - Programming manual made with Sphinx in-repo needs content

We had decided not to move forward with other ports until the modeller webservice was fully functional. It is now, so we're moving on to Muscle and Clustal Omega. Unfortunately I never figured out a way to coerce Python 3.6 to generate stdout and stderr unbuffered or line buffered without resorting to `shell=True`, so we've switched to counting output files instead. This has its own benefits such as a smaller payload than `stdout.txt`.

I have a PR pending on the ChimeraX repo that implements the client side code from both of them and Eric has been on the backend.

Webservices and webservices-test now run in their own virtual environments. Greg supplied me with the WSGI file that he uses for the toolshed, and I was able to adapt it to get both instances of webservices mostly decoupled from the system Python. I may modify `/etc/init.d/rq_workers` to call the virtual environment's version of `rq` – right now I add the virtual environment to the system `rq`'s path – but the need has not presented itself yet.

We have also solved many permissions errors around logging, but one remains: `os.mkdir` is not respecting `setegid`...

I would say right now that the Job API is stable. Currently, unfortunately, the best available documentation is my previous presentation on it, but as we wind down the larger issues of feature development and bug squashing I will put in the time to write official documentation.

We still need to find the time to meet and talk about AlphaFold user quotas. I mentioned before I thought we could use ChimeraX UUIDs to track resource usage, but we should hammer down details.

Aim 3: Qt6 Migration



Status: Standing By for Updates

- Qt6 is more stable than expected!
 - Switched to Qt6 in the daily builds about 19 days ago on 11 March
 - 525 downloads of the daily builds since
 - 7 open tickets regarding Qt6
- At least two have issues that require upstream patches
 - Qt6 macOS support is the same as it ever was
- Hopefully Greg's investigation of flatpak reveals that Qt6 is viable for CentOS 7

The one-liner I wrote for this has spawned a new sport I call “shell javelin”

Aim 4: OpenGL-Vulkan Migration

I think the consensus is still that we're going to deprioritize this effort, but to our collective horror Tom noticed an arm64 OpenGL crash last week. CZI is a two-year grant, so hypothetically we have up to 3 more releases after 1.4 in which to fulfill this requirement. I think we should collect crash data from 1.4 users, evaluate whether it's worth doing based on that data, and if we notice more arm64 crashes perhaps milestone it for 1.6 or 1.7

Aim 5: arm64 macOS Port



Status

- Some libraries still missing arm64 wheels:
 - imagecodecs
 - pytables
- NetCDF4-python wheel broken in 1.5.8 ([source](#))
 - arm64 support confirmed in 1.6
- Tom has made huge contributions
 - OpenMM 7.7 for arm64 macos
 - Confirmed Ambertools20 works through Rosetta

We are very close.

Next week will begin investigating making contributions to imagecodecs and pytables to support arm64

We're within spitting distance of being able to do an arm64 port. I think 1.5 is a reasonable timeframe. NetCDF4 confirms they'll support arm64 in the next release. Tom put in great work to get OpenMM and Ambertools20 working. However I think it may be time to begin investigating making contributions to imagecodecs and pytables to support arm64, so my plan is to work on this tomorrow and next week.

Aim 6: PyPi Package

In the CZI grant we wrote:

To maximize community reuse of ChimeraX code we will distribute a library through the standard Python package repository PyPi with a permissive open source license. Currently we distribute the open source ChimeraX application via our lab website.



Architectural Pathways

- What we wrote in the CZI grant can be interpreted in two ways
 - ChimeraX distribution that does not include the GUI bundle
 - Distribution of ChimeraX bundles as libraries
- Each path forward presents architectural challenges
- `--nogui` distribution:
 - Full ChimeraX release without UI bundle
 - Need to guard code that uses GUI against `session.is_gui` and focus on improving `nogui` mode
- Library distribution
 - Need to guard code that assumes a gui *or* a session
 - Decouple important “business logic” code from session (`if session is None`)
 - Still write to session to support ChimeraX GUI but only if session provided
 - Could invert how we think about ChimeraX: ship an open library but a licensed GUI to take advantage of it
- Open multi-platform questions: Conda vs. PyPi? Conda *and* PyPi?
- Library distribution implies `--nogui` if the user downloads core and every bundle

To me it seems like we can interpret this statement in one of two ways.

Decouple: Perhaps they want to run some useful calculation, but they don't care about ChimeraX's GUI and just want the logic or to call into a service

Last line: So I suppose the real difference between them is intended use case



Necessary Updates

- Webservice port is a soft blocker for PyPi packaging
 - We **must** drop suds-community to update setuptools
- distutils deprecated in Python 3.10 – must update affected bundles to setuptools
- setup.py is deprecated – must update bundle builder
- Other updates depend on our goals

4 step plan for `--nogui`

- Guard gui code, test nogui mode, build without ui bundle, release all at once

3 step plan for library :

- Guard gui and session code, record bundle dependencies, publish bundles as they are updated



Optional Updates

We have to update the bundle builder to handle setup.cfg at least, but there's also pyproject.toml

- setuptools' setup.cfg functionality being absorbed into pyproject.toml
- Same idea as setup.cfg, but other tools can write arbitrary metadata to it
- Each tool simply ignores irrelevant sections of pyproject.toml
 - Deprecate bundle_info.xml, let bundle developers store all info in one file
- Supported by: yapf, flake8, pytest, coverage, flit, setuptools, poetry, mypy, mspyls, pyright, sphinx, tox, etc...
- Should probably integrate some kind of continuous delivery into this project
 - Automated cloud builds on GitHub Actions, etc.
- PyPi packaging is a good project to toss yak shaving ideas into but it's not a blank check.
 - What encourages community contributions?
 - What's easy for bundle developers?
 - What increases our velocity?

I opened a ticket about pyproject.toml just to open a discussion about using it. We're already using it, just not for builds.

TOML is basically a souped up INI file which has its own drawbacks, but the rest of the Python community is standardizing on it.

Lastly, PyPi packaging is a good project to toss yak shaving ideas into because its scope is almost totally definable by ourselves, but I'm also fully aware that it's not a blank check under which we, and mostly I mean I, can sweep interest projects. I mainly want to ask these three questions: ... with a bias towards what leans on native tooling as much as possible.

Thank you!

As always, thanks for giving me the opportunity to present and listening to me. I look forward to continuing to work on these projects and others.