Timeline (one box for each year)

Implement REST interface in Phenix and ChimeraX (including ISOLDE)	Implement interface to fragment extension and ligand placement algorithms	Implement interface to refinement restraints and validation algorithms	Community testing and feedback on Phenix/ChimeraX and iSOLDE interfaces	Implementation of additional interfaces to Phenix algorithms based on community feedback
First				

- “Open in ChimeraX” button
- Center on xyz coordinate (use Christophers code for centering)
- Demonstration for fit_ligand (Tom G. and Zach)
- Billy: Flask prototype (Python 3)
- Phenix: which code to use for ligand-fitting: consolidate
- Fit_ligand: new copy of ligand (as flexibly fit)
- All integration code is in Python3

Unique ID for implementation (Coot: one user updates coot of other user).

Tristan temporary directory clean up code:
https://github.com/tristanic/chimerax-phenix/blob/c7c39333fff6c1e698e04dcc43260c79fd0a93ee/src/rest_server/phenix_side/server.py#L319

ChimeraX REST API
https://www.cgl.ucsf.edu/chimerax/docs/user/commands/remotecontrol.html

What do we want from the communication protocol?

- Do we need a client-server relationship?
  Pros:
  - If the job is run on another machine: network capability including encryption/compression would be useful). If we simply run the command line tool (such as Molprobity), then results can only be obtained by parsing log files or json files.
  - Protocol can be generalized.
  - Can run in a temporary directory (don’t clutter your working directory)

  Cons:
  - Program template can create json result files.
  - OK to parse if we only want to implement integration for a few programs
  - OK if each GUI is custom made

- wish-list for integration:
    - map_symmetry
    - ligand_placement
    - combine_focused_maps (needs models fitted into maps)
    - real_space_refine
    - phenix.refine
    - comprehensive_validation
    - ReadySet?
    - autosharpen
    - density_modification_cryoem
- Needs to be available on all three platforms (Mac, Linux, Windows)
- Python 3
- Ideally neither program would have to have knowledge of the other program's source code
- Things Phenix would ask the viewer to do:
    - Show model
    - Show maps
    - Center (without optimization of the view). Center using selection or xyz?
    - Show markup

What are the possibilities? Make a survey. (Zach, Billy)

Organized by transport method

- HTTP
    - XML-RPC
    - JSON-RPC
    - Pyro5
    - Could bind a small webserver to localhost and expose an API
        - waitress, gunicorn, hypercorn, etc.
- Sockets
    - `socket(AF_UNIX, ...)`
    - `socketserver.UnixStreamServer(...)`
- Pipes
    - Named pipes -- `os.mkfifo`
- Shared memory
    - Requires python3.8 -- `multiprocessing.shared_memory`
    - The fastest form of IPC
    - Open interface for other programs?
- Message Queues
    - ZeroMQ (abstracts sockets, but not communication method)
        - Have to consider cross-platform availability (Windows, primarily)

Billy:

- REST API frameworks:
    - Flask (popular, pick and choose): python. Comes with its own basic server.
    - Django (no, too complicated)
    - Falcon (smaller community): python

- websockets?

What Phenix / ChimeraX interaction capabilities do we want to release in 6 months?

- Show and update models/maps from Phenix in ChimeraX (“open in ChimeraX button”)
- Demonstrate that ChimeraX can launch a Phenix job (phenix.refine or phenix.real_space_refine)
  (ISOLDE can write phil files = settings file for Phenix. Step further: launch a job or launch the GUI)

Pyro server or REST?

Can Pyro be used for Coot or Pymol?

Coot:

- uses Python 2, but release 2022 will use Python 3.7
- uses xmlrpc.

Pymol:

ChimeraX:

- Some xmlrpc is available

xmlrpc is a bit limited

What modules does python 3.x have for communication?

Launch Phenix jobs via REST, not the command line.

About Pyro5

- https://pyro5.readthedocs.io/en/latest/tipstricks.html#pyro-via-http-and-json
- Can we run pyro5 just on the server side? If both: maybe not so useful for us. → doublecheck
- Will pyro5 be around in 5,10 years? Is it established/supported? We need something that will be maintained. → evaluate
- Has compression/encryption

Should anything go into the October 2021 Phenix release?

No.

Should anything go into the October 2021 ChimeraX release?

No

Should we only develop the ChimeraX interaction for Phenix Python 3 or also Python 2?

Only Python 3

Are Coot and PyMol and ChimeraX expected to all offer the same API for Phenix to use? Probably not. So Phenix tools will need code specific to each graphics program? Or will they query if the graphics program offers the capability they want to use?

Not the same API. Phenix expects all functionalities to work in all viewers. (no viewer specific behavior)

How does Phenix cope with tasks that are only available in certain viewers? → Phenix basically just asks to center. The user then does whatever can be done with the particular viewer. So Phenix will not launch any functionality in the viewer.

Markup: MolProbity, ISOLDE. ChimeraX markup style is different from Molprobity markup. Exact styles are currently not there (but can be added).

Things Phenix would do:

- Show model
- Show maps
- Center (no optimization of the view). ChimeraX: don't use xyz, center on residue via selection (would need a "translator" for selections in Phenix and ChimeraX)

For clashes: Phenix uses xyz to center on the clash. Instead, maybe it would be useful to center on the two clashing parties.

- Show markup

What version compatibility do we want to require between Phenix versions and ChimeraX versions? How will the programs warn if the versions are incompatible?

Which version of Phenix is incompatible with a version of ChimeraX?

→ Could put a warning "upgrade your Phenix" if versions are incompatible. Could be done with communicating API.

But: don't force users to do something. Try avoiding incompatibilities.

Does every Phenix tool start a new instance of ChimeraX? Is that what Phenix does for Coot and PyMol?

Does not open a new coot. Maybe: same for pymol (double check).

Also: What do users want?

Does it make sense to pass new scenes from Phenix to ChimeraX and have ChimeraX update everything? Or should there be specific API methods like "focus_on_residue()", "update_model_coordinates()", ...

Use residue selections, not xyz. Replace old model with new model.

API: ~dozen functions with limited number of arguments; don't make it too complicated (could make it more restrictive; if based on building blocks, can always make it more complicated in the future, if we want to)

We should make some example code for how ChimeraX calls Phenix with Pyro for a ligand fit computation.

Have an example for ChimeraX → Phenix direction. But not sure yet if we'll use Pyro.
Need server/client on both sides. Pyro can do compression/encryption.

What is the format of a Phenix selection string? Will ChimeraX be able to easily parse that in the scene data structure?
https://phenix-online.org/documentation/reference/atom_selections.html