Fragalysis - Job Launcher Phase 2

Author: Duncan Peacock
Version: Approved
Date: 05/05/2022

Release History

Version	Contents	
0.1	Initial discussion document - will likely change significantly.	
0.2	Updates following meetings on 13/04/2022 and 19/04/2022. - Results sdf to Fragalysis conversion - Optional delete of computed set following upload. - Improvements to compound set processing.	
0.3	Updates following meetings on 28/04/2022 - Select/transfer of RHS .mol files as part of data. - Appendix detailing download options for RHS	
1.0	Design agreed in meeting 05/05/2022. Any updates will be done in a new version 1.1	

Design Task:
https://github.com/m2ms/fragalysis-frontend/issues/839

Epic Task
https://github.com/m2ms/fragalysis-frontend/issues/850

Introduction

https://github.com/m2ms/fragalysis-frontend/issues/791 outlines high level requirements for running jobs in the Squonk 2 ecosystem from the Fragalysis frontend.

This document outlines the work required to the Fragalysis frontend (React Viewer), Fragalysis backend (Database and API) and Squonk2 for the phase 2 work to be able to launch and track Squonk2 jobs from Fragalysis covering mainly the changes/improvements to the compound set functionality to be able to upload results sets from squonk back into Fragalysis. It also covers some follow-up technical debt that should be resolved.

The phase 1 design is here:
https://docs.google.com/document/d/1JsfzqstUOZ4F6v4QIiq2OkrOrwZWYXSBr4TZtrphPxI/edit#heading=h.yfjku6idmcje

Terminology

Squonk2 also has a UI frontend (React) but with a NextJS server side component and a Flask-based backend known as the Data Manager. For clarity I've stuck to Squonk2 throughout this document.

High Level Changes

Job Launching:

1. The file transfer process will be expanded to allow molecules (in the form of the .mol format) from the RHS to be selected and added to the transfer, based on the name of the computed molecule.
2. Once a job has finished then, if the job has been configured for the automatic upload of results (of a compatible SDF file), the Fragalysis backend will call Squonk2 to retrieve the results file(s), add any metadata from the job and upload them into Fragalysis using the Computed sets upload process.
3. If the *job_request* record is subsequently queried from the frontend and a compatible (SDF) results file has been uploaded then additional information on these new compounds can be provided to the frontend.
4. The frontend can then regenerate the snapshot with the results compounds from the SDF selected to show the differences (e.g. in the 3D viewer) and/or *optionally delete the computed set if it is not wanted*.

SDF File format:

There is a significant difference between the SDF files returned from Squonk and those currently uploaded into Fragalysis as Computed sets. The computed set SDF has a version number, a "blank molecule" containing descriptions of fields and required parameters.

The instructions for the compound set upload are described here. Current version is 1.2 (last post in the link): https://discuss.postera.ai/t/providing-computed-poses-for-others-to-look-at/1155

To fill these parameters when running a job, a new process will be developed that will run alongside the existing process (which must be kept). The parameters in the current compound set SDF file can be broken down into three groups and these will dictate how the new process will work:

- Group 1: Identification and Submission Information - Who ran what job where. These fields are mostly provided in the "blank" molecule in the SDF file and are used to update the *ComputedSetSubmitter* model information. In this case the data to fill these already exists in Fragalysis - it has been created as part of the job launch process.
- Group 2: Fragalysis Reference Molecules (ref_mols). The information in this parameter originates in Fragalysis but also depends on the job. The Fragalysis protein id (e.g. x0034_0A) will be added as the title line of the .sdf and .mol files that are transferred to Squonk (note: it would be preferable if these were already added by XCR). When the job runs the ref_mols parameter will be added to the output results file based on the job/options selected.
- Group 3: Job Specific Parameters - new fields that are generated by the job and are currently specified in the "blank" molecule in the SDF file. These parameters will be created by the squonk job. These parameters will be communicated back to Fragalysis in a simple json file alongside the results file and downloaded from the same location in Squonk (e.g. if the results file is called "results.sdf" then there will be an associated file called "results.params.json").

With these changes, the current Combined set SDF *could* be logically constructed as it is now. However in practice it is probably more maintainable and simpler to develop a simplified validation process alongside the existing file-based process -> The results feeding into the current process step.

Further information on the groups and how to fill the fields is given here

Notes/Limitations:

- Following the changes to the existing compound set processing (see below), some fields in the existing process could also be automatically generated (e.g. through the Login) . It could be possible in future to simplify the existing “blank” molecule - or remove it altogether. However this is out of scope for this change. With the exception of the changes described in the next section, the existing process will continue to work as now.
- It is also possible to upload a (number of) PDB files in the current Computed set processing. We still need to work out how/whether this can be used.
- Note that ref_pdb and original smiles may not be possible at this stage.

Compound Set Processing

1. The existing computed set functionality must be changed to accept the resulting SDF files from the Squonk jobs as described in the “SDF File format” discussion above.
2. The existing computed set screens/functions (update/upload_cset) will be changed to be for logged-in users only (like for upload_tset).
3. The existing computed set screens/functions (update/upload_cset) will be modified to allow deletion of existing computed sets for logged-in users. This will allow the computed sets to be managed following the upload. On the upload and update cset pages, anonymous computed sets will be visible, but you can only update/delete sets that you own. A delete button will be added to these pages so you can select a dataset and delete it. This will involve adding the *user_id* to the ComputedSet model. Current computed sets will be updated so that they are attached to the anonymous user (part of the migration).
4. As the upload functionality will now be called from both the existing screens and the new job launching some improvements/corrections will need to be made to the error checking during the processing and email notification will be added to success/failure of the upload operation on the compound set.
5. In addition, the existing computed set API will be changed to allow DELETE - this is so the front end can also provide a delete button so that logged-in users can simply remove computed sets for jobs they have just run but no longer want.
6. A new “download Compound Set” api will be introduced that can also be called from the front end. In the first phase, this will just be a “get” on the file that is stored in the upload process (in the computed_sets subdirectory in the media).

Limitations from Phase 1 in Scope for Phase 2:

1. In phase 1 for simplicity the *squonk_job_config.yaml* contained the complete configuration for the test job. However ideally the definition for a job could be retrieved from Squonk2 directly and then updated with Fragalysis-specific options. This will be included in phase 2. The *squonk_job_config.yaml* will be simplified to contain the Fragalysis-specific options and a simple

API will be introduced to the backend to return the squonk configuration (which will be obtained directly from Squonk by the backend using the squonk api).

2. In phase 1 there was no check on duplicate job requests. It seems this is straightforward to add now so will be included in phase 2.

3. One issue arose during testing in phase 1 that must be addressed in phase 2: The results filename is specified as an option on input and this caused a few problems: Multiple users can run jobs in the same squonk project workspace. If two different users use the default filename from the options (e.g. “results.sdf”), then it causes the second job to fail as it cannot overwrite the results from the first run. Overwriting these results is also wrong of course. Furthermore, in phase 2 the results sets that will be uploaded into fragalysis are supposed to have filenames of format: compound-set_<name>.sdf - and this name must be unique within fragalysis (currently limited to 50 characters).

   In discussions around the design there was a strong preference for having a default opinionated directory and filename, for easy tracking/usability. Hence to solve this problem, I propose that the results files are specified to be placed in a directory of format:

   `<username>/results_<job_name>_<date (to second)>.sdf.`

   On upload to Fragalysis (if specified), results_<job_name>.sdf will be renamed to be:

   `compound-set_<job_name>_<date (to second)>.sdf.`

   The results filepath and filename are defined in the job specification at run time so this must be defined on the Frontend

Limitations from Phase 2:

1. From Phase 1: In phase 1, only the .mol, .sdf and _apo_desolve.pdb files were transferred to Squonk 2 for a snapshot. This should be sufficient for the initial list of jobs required by Fragalysis.
2. In phase 1, we initially implemented a *default_squonk_project_id* for each target so all jobs for this target will run under the same Squonk project. Further changes to this processing are out of scope for phase 2, but we will need to extend this when private projects are requested. Also need to consider whether users want to run jobs in their own project rather than one shared by all users of a fragalysis target.
3. Download of Computed sets. This will be kept simple in Phase 2 by allowing the option to download the complete computed sets (i.e the ones stored from the upload process). There will be no additional processing to, for example, split the SDF files into individual molecules.
4. Upload of Computed sets. It will not be possible to upload results sets from squonk via the current process (i.e. the *viewer/upload_cset*) page as for backwards compatibility this will have to follow the current process.
5. There is no functionality at the moment to change the owner of a computed set once it has been uploaded. This should be a relatively straightforward change later depending on whether this is wanted on the react frontend or backend. At the moment this would need to be done manually, but is not expected to be a major issue at this stage.

6. There has been some discussion about whether transferring files to Squonk is the right approach. It is possible that a cleaner approach in future might be for Squonk to “pull” files from Fragalysis when it needs them. In most cases the API’s already exist and are ISPYB safe.
7. There was also some discussion on leaving physical results in Squonk and just providing links to them in the Fragalysis database. I would not recommend this as it would cause complications in the backup/restore of data (and transfer to different systems) and whether files would be available to users who have had ownership transferred to them. An uploaded physical results file should reside in the media directory so it is backed up at the same time as the database data that depends on it.

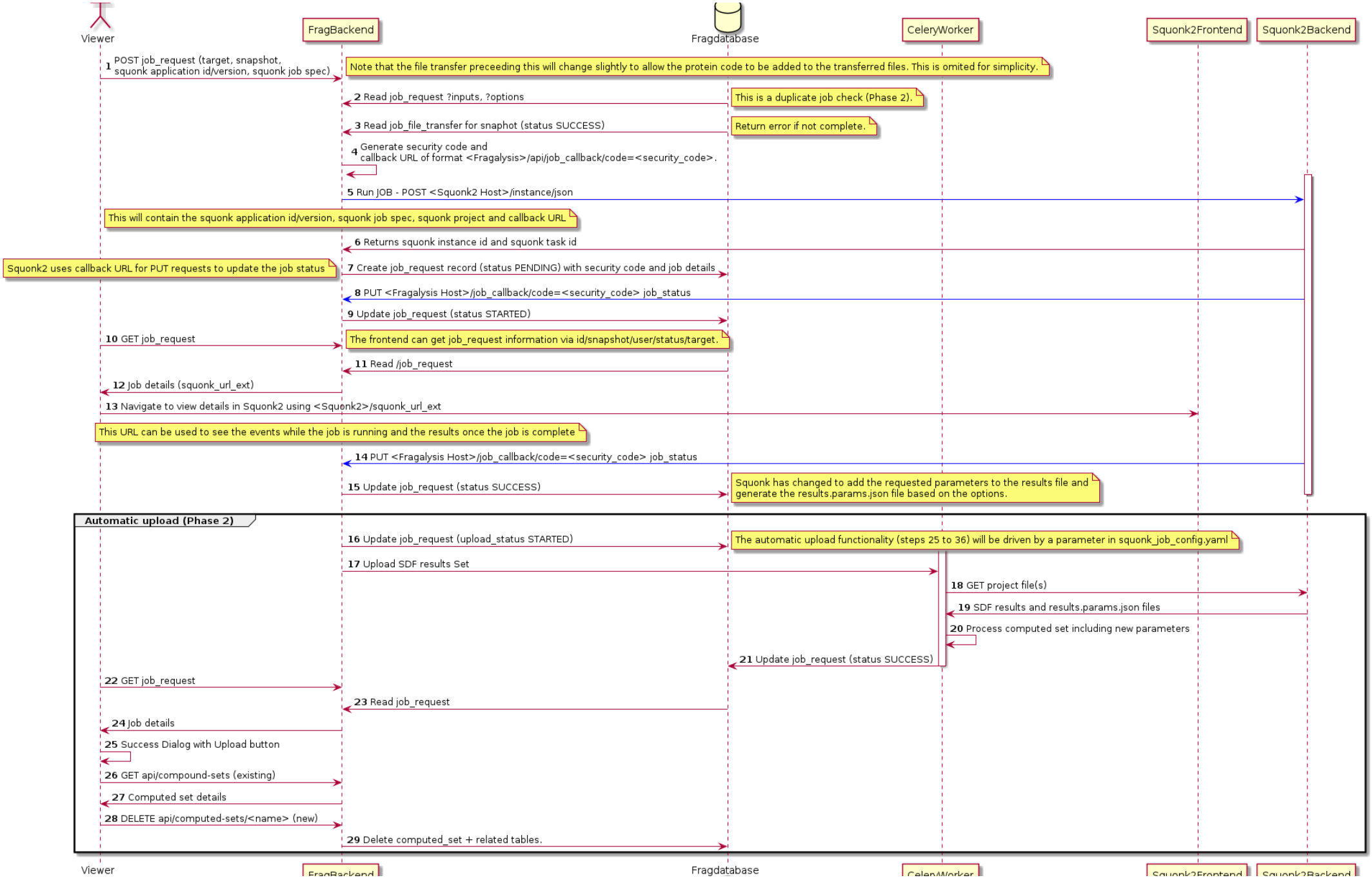
Use Cases:

UC	Description
1	Job Submission: - User requests job - Squonk2: Call Backend to update job record job status SUCCESS. - Backend: Backend: (not phase 1) If applicable, initiate automatic upload of results SDF file from Squonk2 into Fragalysis. - Frontend: check job status - receives status of success. - Backend: Returns details for job including ability to jump to Squonk2 to view job execution details. - Frontend: (not phase 1) show Success dialog. User selects to “see results”. Results snapshot created and user redirected to uploaded molecules in 3D viewer. - (optional) - User requests to delete uploaded compound set.
2	User attempts to queue the same job more than once - If the job is processing or finished, job details will be returned and a URL to the processing/finished job details will be provided. - If the job has not yet started (for example, waiting for files to transfer), then an error message will be returned.
3	Delete computed set: - User logs on and navigates to the upload_cset page (details tbc) - User can view sets that they own and anonymous sets. - User can delete computed sets that they own.
4	Add computed set: - Will now require login - Will return an email indicating success/failure
5	Job Submission (other than fragmenstein combine) - Expanded selection of jobs available (enabled by improved process for job configuration). - Possibility to select computed molecules from RHS to be transferred to Squonk
6	Computed Set Download

Solution Outline

Job Submission Process Flow (UC1)

The figure below shows the updated job submission process where it relates to the automatic upload of results.


Viewer
FragBackend
Fragdatabase
CeleryWorker
Squonk2Frontend
Squonk2Backend
1 POST job_request (target, snapshot, squonk application id/version, squonk job spec)
Note that the file transfer preceeding this will change slightly to allow the protein code to be added to the transferred files. This is omited for simplicity.
2 Read job_request ?inputs, ?options
This is a duplicate job check (Phase 2).
3 Read job_file_transfer for snaphot (status SUCCESS)
Return error if not complete.
4 Generate security code and callback URL of format <Fragalysis>/api/job_callback/code=<security_code>.
5 Run JOB - POST <Squonk2 Host>/instance/json
This will contain the squonk application id/version, squonk job spec, squonk project and callback URL
6 Returns squonk instance id and squonk task id
Squonk2 uses callback URL for PUT requests to update the job status
7 Create job_request record (status PENDING) with security code and job details
8 PUT <Fragalysis Host>/job_callback/code=<security_code> job_status
9 Update job_request (status STARTED)
10 GET job_request
The frontend can get job_request information via id/snapshot/user/status/target.
11 Read /job_request
12 Job details (squonk_url_ext)
13 Navigate to view details in Squonk2 using <Squonk2>/squonk_url_ext
This URL can be used to see the events while the job is running and the results once the job is complete
14 PUT <Fragalysis Host>/job_callback/code=<security_code> job_status
15 Update job_request (status SUCCESS)
Squonk has changed to add the requested parameters to the results file and generate the results.params.json file based on the options.
Automatic upload (Phase 2)
16 Update job_request (upload_status STARTED)
The automatic upload functionality (steps 25 to 36) will be driven by a parameter in squonk_job_config.yaml
17 Upload SDF results Set
18 GET project file(s)
19 SDF results and results.params.json files
20 Process computed set including new parameters
21 Update job_request (status SUCCESS)
22 GET job_request
23 Read job_request
24 Job details
25 Success Dialog with Upload button
26 GET api/compound-sets (existing)
27 Computed set details
28 DELETE api/computed-sets/<name> (new)
29 Delete computed_set + related tables.
Viewer
FragBackend
Fragdatabase
CeleryWorker
Squonk2Frontend
Squonk2Backend

Steps:

Pre-condition, the files transferred to squonk will have been modified to contain the fragalysis protein codes as parameters. This will apply initially to the .mol and ,sdf files. For the .apo_desolve.pdb files, we are not yet certain how this would be done - it may be better to do this in XCR. Also, it will now be possible to add compounds from the right hand side (i.e. from computed sets). See here for more details.

1: To launch a job, the job is selected on the Job launcher pane as for phase 1. This will open/extend the Job launcher pane with options based on the Squonk_job_config.yaml file *(which will now contain Fragalysis specific parameter) and the results of the call to the job_spec API which gets the job specification from Squonk*.

2: Fragalysis will check that the same job has not been queued before (by checking the job_request table using the job specification.

3 - 15: Restates the phase 1 processing for clarity. As part of the job processing, Squonk will add parameters to the output sdf file as specified in the options. These added parameters will be listed in a new file: <result filename>_params.json.

16 - 21: Once a job has finished (the job_status has been updated to SUCCESS) then, if the job has been configured for the automatic upload of results (of a compatible SDF file), the Fragalysis backend will call Squonk2 to retrieve the results and <results>_params.json file(s) and add any metadata from the job (e.g. PDF or fragments) and upload them into Fragalysis using a new process that will make use of the existing computed sets upload functionality. This is another async task that will be tracked by a second status field in the job_request table called *upload_status*. The status will proceed from PENDING to STARTED to SUCCESS.

Notes

- The resulting SDF files will be translated to be compatible with Fragalysis computed sets.
- The process will be automatic - but the user will easily be able to delete the file via a new API .
- The computed set upload process will be updated to improve its error checking and stability where possible.

22 - 24: If the *job_request* record is subsequently queried from the frontend, then a summary of the results can be provided. Furthermore, if a compatible (SDF) results file has been uploaded then additional information on these new compounds will be provided.

25 - 28: The frontend can then with the results compounds from the SDF selected to show the differences (e.g. in the 3D viewer). This should be possible via existing functionality (i.e. the *computed-sets* API).

29 - 30: The existing computed set functionality (*computed-sets* API) will also be changed to allow deletion of existing computed sets for logged-in users. This will allow the computed sets to be managed following the upload. Alternatively, imputed sets can be deleted via the separate "*viewer/upload_cset*" page.

Mapping Squonk Results Files to Fragalysis Computed Sets

As stated earlier, to make it possible to upload a squonk result file to Fragalysis a mapping must be made to simulate the parameters uploaded in the so-called “blank” molecule in the computed set upload process. The parameters can be split into three groups:

Group 1: Identification and Submission Information

This will be generated in Fragalysis as follow:

- *submitter_name* - filled with Job submitter user name
- *submitter_email* - filled with Job submitter user email
- *submitter_institution* - Set to “unknown”
- *generation_date* - Job run datetime
- *method* - Job name
- *ref_url* - URL to results page in squonk (contents of *squonk_url_ext* in *job_request*)

Group 2: Fragalysis Reference Molecules

The *ref_mols* parameter is required in the computed set upload and is a comma separated list of the fragments that inspired the design of the new molecule (codes as they appear in fragalysis - e.g. x0104_0,x0692_0)

To provide this information, the process will be as follows:

1. The Fragalysis codes will be provided to Squonk as parameters in all the files transferred across to Squonk (.mol, .sdf and .apo_desolve.pdb) as part of the snapshot files transfer process.
2. Squonk will then add a new “include-ids” input option to define the applicable input protein codes and add these as a comma separated string parameter in the output SDF file. The name of this parameter will be stated in the associated parameter config file (see below).

Group 3: Job Specific Parameters - new fields that are generated by the job.

These parameters are defined in the Squonk job configuration. When the job runs, Squonk will add the parameters to the SDF results file. The names of the parameters will be added as simple name=value pairs to a new JSON file that will be created alongside the SDF results file. One record will always be present in this file - the *ref_mols* parameter described above.

For example if the results file is created in:

```
<username>/results_<job_name>_<date (to second)>.sdf.
```

The parameters that have been added will be located in:

```
<username>/results_<job_name>_<date (to second)>_params.json.
```

Adding RHS Compounds to Input Parameters

For one of the initial jobs required by Diamond it will be necessary to have the ability to transfer over computed molecules also from the right hand side. This functionality will be added but should kept as simple as possible at this stage. It can be made more sophisticated later. The process will be added through a simple extension of the *job_file_transfer* processing introduced in phase 1.

1. The job_file_transfer api will have a new input field “Compounds” added to it (this is already in the backend table).
2. A selection process will copy computed molecule identifiers from the RHS (e.g. PAU-WEI-b9b69149-9, DAR-DIA-caba39e3-12) and place them in the new field as a comma separated list. These correspond to the name field in the ComputedMolecule model.
3. The list will be processed in a similar way to the list of Protein codes. For each identifier, the “.mol” file will be extracted from the computed molecule (sdf_info field) and transferred to Squonk in a file prefixed by the name (e.g. PAU-WEI-b9b69149-9.mol). This computed molecule name should also be added as a parameter in the .mol file in a similar way to how the protein code will now be added.

Note, that whether these molecules are actually added to a snapshot falls outside the scope of this document. The codes are present in the *job_file_transfer* table connected to the snapshot and this is all that is necessary for the job to work.

Changed Required By System

Fragalysis Frontend

Changes described here will only refer to basic functionality to clarify the flow of information. Look and feel is out of scope of this document.

To enable the backend changes to work as expected, the Fragalysis frontend will need to handle:

1. Change to get squonk job specification from new *job_config* API and merge with *squonk_job_config.yaml* information.
2. Add ability to select RHS compound and add name to comma separated line in *job_file_transfer* API.
3. Default the job results filepath and filename to prevent results files being overwritten as described here.
4. Display uploaded compound set in pose viewer.
5. Ability (through a button etc.) to call *api/compound-sets/<name>* to delete an unwanted compound set after display.
6. Ability (through a button etc.) to call new *api/download_compound-sets/<name>* to download a compound set.

Note that the rest of the functionality that will handle the deletion of the compound sets will be handled by the backend repo (this is how it was originally designed).

Fragalysis Backend

Changes required:

1. New API: *job_config* to get a job configuration from Squonk2
2. File Transfer - Change the processing to add a protein-code parameter to the *.mol*, *.sdf* file types.
3. File Transfer - Add the RHS ComputedMolecules .mol files using the viewer_computedmolecule.name field as identifier.
4. Automatic upload of results files via celery task
5. Improved functionality to the upload_cset computed set functionality. Logged on users will be able to delete computed sets that they own.
6. Change *api/compound-sets* to add deletion functionality
7. New api to download a computed set that has been uploaded. This will download the file from *viewer_computedset.submitted_sdf* in a similar way the how the existing *api/protpdb* API downloads PDB files for proteins.

Note:

There is **no** intention at this stage to create something like the download_structures functionality for computed sets. The download will be kept simple to add the functionality cost effectively.

File Transfer

Add parameters to existing files_types.

Change the processing to add a *ref_mol* parameter to the *.mol*, and *.sdf* file types. Table *viewer_protein* already contains this code as it's validated and stored in Fragalysis as part of the upload and used to index the file transfer.The ref_mol can be defined as the (first part of) *viewer_protein.code* with the target stripped out. Hence: For protein code CD44MMA-x0022_0A, the *ref_mol* is 'x0022_0A'.

Add Computed Molecules from RHS.

As described here, add the processing for RHS molecules. A refresh mechanism similar to the one built for the proteins phase 1 will be used to check if the RHS molecules have been updated when a job is queued (based on the computed set *upload_datatime* described in the computed set improvements below). In this case, the file will be transferred again. The ref_mol can be defined as the whole *viewer_computedmolecule.name* (e.g. PAU-WEI-b9b69149-9)

Automatic Upload

After the SUCCESS state has been received by Fragalysis, the job request parameters will be consulted to see whether an automatic upload has been requested.

In the case of an automatic upload:

- The *upload_status* will be changed to ‘STARTED’
- A new celery task will be created that will GET the configured results (and parameters) files from Squonk2.
- The functions within the existing computed set Celery functionality will be adapted to also process the results file in Fragalysis as a new computed set.
- On completion of the upload, the status in the job_request will be changed to “SUCCESS” and the table updated with a reference to the computed set.
- It is recognised that there are shortcomings with the existing celery tasks. Within the limits of the budget, the validation processing will also be improved - specifically if there is a script failure then there should be a backup process to mark the upload as ‘FAILURE’ and provide a method whereby a new table will contain the reason for the failure (in the event that it is nor picked up by the existing validation processing).

Computed Set Deletion and Improvements

The existing computed set screens/functions (update/upload_cset) will be changed to be for logged-in users only (like for upload_tset). If the user is not logged on, a message will be displayed inviting the user to go to the landing page to logon.

Now that users must be logged on the *cset_keys* functionality, that was introduced to provide authentication keys for upload, will be removed from both screens as it’s no longer required. It’s currently mostly commented out and so not actively used in any case.

In practice both of these pages are similar in how they are used. The main difference is that the update_cset page is aimed at adding an SDF file to an existing set - so the “blank” molecule is not required.

Both pages show a list box of existing computed sets. The contents of this box will be modified so that it shows the computed sets for anonymous users and the logged in user.

Next to the existing SUBMIT button, a DELETE button will be added. You can only update or delete computed sets that you own. If I can work out how to prevent the anonymous computed sets from being selected in the list - I will do this, but otherwise if the user selects an anonymous computed set and presses SUBMIT or DELETE, an error message will be returned.

Pressing DELETE will result in the computed set being removed from the database. This is already configured using a cascade delete and seems to work OK by just deleting the high level task. There is no need for a new celery task in this case.

The changes above involve adding the *user_id* to the ComputedSet model. Current computed sets will be updated so that they are attached to the anonymous user (part of the migration).

Validation Improvements

As the upload functionality will now be called from both the existing screens and the new job launching some improvements/corrections will need to be made to the error checking during the processing and email notification will be added to success/failure of the upload operation on the compound set.

Note that the code here is complex and has caused several problems in the past. A rewrite is out of scope, however some improvements will be made. These are listed below:

1. Currently, there is no automatic notification of the success, validation error or failure of an upload. You have to either remain on the page or query the task via a separate page. As the user will now be logged on, an email will be sent notifying the user of the final status of the upload.
2. The validation will be improved to check lengths of some name fields that are limited in length (such as the computed set name primary key which is limited to 50 characters).
3. In the case of a crash in the code, currently no message is returned at all if you are not looking at the page.

For 1), some email functionality was introduced for the target upload functionality and this can be simply repurposed in the cases of normal validation errors and a successful return. However in case 3) it also returns no result.

To improve 1) and 3), therefore, a few improvements are proposed:

- Add also the fields *upload_task_id*, *upload_status* and *upload_datatime* to the computed set functionality. These will be updated as the celery task is progressing (in a similar way to the job_file_transfer and the target).
- Add what is effectively a CRON job (using the Django Celery Beat library to run every 5 minutes and check the *upload_task_id,* update the *upload_status* and *upload_status_datetime* fields and send a notification accordingly. This will also pick up the situation where the python has failed completely. In this case it is proposed that in addition to the status update, a new table is written to - viewer_task - that contains the traceback of the failed celery task to aid data correction and debugging.

Note:

Ideally, at some point in the future, this upload_cset and upload_tset functionality should be changed to be called from a pop-up window on the Fragalysis frontend, but it was not designed that way and major changes to this area of the UI are out of scope for this change.

Data Model

Viewer_computedset (existing)

- *owner_user_id* (FK): This is the id of the user who uploaded the computed set
- *upload_task*
- *upload_status*
- *upload_status_datetime*

Viewer_jobrequest(existing)

- *Job_status_datetime* - This was missed in the original implementation, but should be corrected.

Viewer_task (new)

- *celery_task_id*
- *status*
- *status_date_time*
- *message*

Configuration tables in order to install the Django Celery Beat library

API Changes

API *job_spec* (New)

GET

Parameters:

- Job name

Returns the job specification from Squonk for the requested job name.

API *compound_set_download* (New)

GET

Export file from *viewer_computedset.submitted_sdf*

API *job_file_transfer* (New)

POST

Add new compounds comma separated string of *viewer_computedmolecule.name*(s).

GET

Return compounds as JSON (similar to proteins).

API *compound-sets* (Changed)

DELETE

Parameters

- name

Add deletion functionality
This will only be possible for logged-in users
The user can only delete computed sets that they own.

Squonk2

Changes required:

1. Add processing to always create the *ref_mols* parameter from the input files and job configuration.
2. Add processing to create the *results_<job_name>_<date (to second)>_params.json* from the job annotations to match the parameters that are added to the results sdf file..

Non-functional Requirements

Environment and Process Changes

All URLS will be relative (if available).

Authentication/Authorization

The user will be logged on to Fragalysis via Keycloak.

Testing and Deployment

No specific data-related steps should be required for this specific phase, but remember that configuration steps are necessary for jobs to run in a new target as this functionality becomes more widely used.

1. Create a target-related project in Squonk and manually update the *viewer_target* table record with the squonk project id.
2. Reload the target data so that the .sdf and .apo_desolve.pdb file references are correct inside Fragalysis.

Database Conversions and Clean up of Old Data

1. Existing computed sets will automatically be given to the “anonymous” user as part of the database conversion.
2. Alterations of ownership (assigning a new user) will have to be done manually at the moment.

Documentation

This document (and the *puml* sequence diagram) will be added to the Fragalysis Backend Repository.

Roadmap

This section contains suggested tickets with rough time estimates. If the scope changes significantly these estimates would have to be revisited:

Ticket	Description	Est. IM	Est. M2M
1	Frontend changes (M2M) 1. Get Squonk job config. Configure results file path/name 2. Handle display of result computed set following upload. 3. Delete and download buttons	1 day	
2	Backend Changes (Duncan) 1. API to return job configuration 2. Changes to computed set functionality to allow deletion of sets for logged in users 3. Kick off import of results sets automatically at the end of a job if so configured.. 4. Translate Squonk results to Fragalysis compound set. 5. Improved validation/error processing 6. Basic API to download a compound set 7. Select from RHS	12 days	
3	Squonk Changes - Ref_mols - Creation of <result>_params.json	2 days	

Notes:

- Estimate for 2) includes introduction of basic automatic tests for compound set upload in a similar way to target set upload.

Appendices

Phase 1 High Level Changes

At a high level (for technical detail see the Solution Outline and following sections) the process will be as follows:

1. A default project for each target will be defined in Squonk2 in which the jobs will be run. This will be configured in Fragalysis.
2. The user logs on (only logged on users should be able to start jobs. The Fragalysis user will also be the user for Squonk2 using keycloak/single sign-on).
3. The frontend creates/selects a snapshot, based on a subset of the molecules (LHS and in future RHS).
4. The frontend calls the backend to initiate an asynchronous copy of the files related to the snapshot from Fragalysis to the Squonk2 project controlled by a new *job_file_transfer* table.
5. To launch a job, the job is selected on the Job launcher pane in Fragalysis. This will open/extend the Job launcher pane by using a *squonk_job_config.yaml* file (provided to the frontend repo by IM with the parameters Squonk2 needs to launch jobs) to flexibly construct an options dialog in Fragalysis using a React Forms library.
6. When the launch button is pressed, Fragalysis will check that the file copy has completed successfully (and in future that this is not a repeat request) and then Squonk2 will be called from Fragalysis (backend) to start the job - the request containing a generated callback URL - and than a job record will be created in Fragalysis to track the job.
7. Squonk2 will use the callback URL to update the status of the job. The job record will also provide the Fragalysis frontend with a Squonk Task-id that can be used to create a URL that can be used to navigate to the Squonk2 frontend to view progress/results.

Compound Set Deletion

Original Solution Roadmap (as discussed with Ruben on 01/12/2022

– back end only is sufficient at this stage.

1. Change update/upload_cset to be for logged-in users only (like for upload_tset). This will probably involve adding the user_id to the ComputedSet model. Current computed sets will be updated so that they are attached to the anonymous user (part of the migration).
2. On the upload and update cset pages, anonymous computed sets will be visible, but you can only update/delete sets that you own.
3. Check the functionality of update_cset page works. Aim is to add a delete boolean to this screen so you can select a dataset and delete it. If not I'll create a delete_cset page – since it won't need the celery task!
4. Should be sufficient to delete the ComputedSet record for the user.
5. Can add the email notification in if it's free (which it is more or less)

Add User to ComputedSet
Make anonymous computed sets available to be seen by everyone – but no delete available for anonymous users.

Compound Set Specification (Version 2)

Specification - ver_1.2

edit: clarification on pdb.zip upload format

The upload format for compounds will be allowed in one of two ways:

A single sdf file
A single sdf file plus pdb files for the ligands to be loaded into in fragalysis.
The sdf files for these two options will have a standardised format, to allow the following options:

The fragments that inspired the design of each molecule can be specified
The protein (in pdb file format) for each molecule can be specified
Any number of ‘properties’ or ‘scores’ can be specified.
The (SDF) format is as follows:

The sdf file name will be: compound-set_<name>.sdf with <name> replaced with the name you wish to give it. e.g. compound-set_fragmenstein.sdf

A ‘blank’ molecule will be the first in the sdf:

This molecule will contain all of the same fields as the sdf, containing a description of those fields.
The 3D coordinates of this molecule can be anything - they will be ignored.
The name (title line) of this molecule should be the file format specification version e.g. ver_1.2 (as defined in this document)
The molecule should have the following compulsory fields:
ref_url - the url to the forum post that describes the work
submitter_name - the name of the person submitting the compounds
submitter_email - the email address of the submitter
submitter_institution - the submitters institution
generation_date - the date that the file was generated in format yyyy-mm-dd
method - a name for the method used to generate the compound poses
NB: all of the compulsory fields for the blank mol can be included for the other molecules, but they will be ignored

Every other molecule in the sdf file will be assumed to be a molecule that is a computed molecule, and should:

Have the same properties as the blank molecule, but with their values instead of description. - for the ref_url field, you can leave this blank, as it will be ignored for molecules that are not the blank molecule
Have a name that is meaningful, and will eventually be displayed in Fragalysis - Use the PostEra submission ID, or if not available, a name that is meaningful (e.g. the PDB code, name used in publication, etc.)
Have the three following compulsory property fields:
ref_mols - a comma separated list of the fragments that inspired the design of the new molecule (codes as they appear in fragalysis - e.g. x0104_0,x0692_0)
ref_pdb - either:
the file path of the pdb file in the uploaded zip file:
Example: If you upload a file called references.zip that contains a pdb file new_protein.pdb, the corresponding path in the ref_pdb file would be references/new_protein.pdb
the code to the fragment pdb from fragalysis that should be used (e.g. x0692_0)
original SMILES - the original smiles of the compound before any computation was carried out

NB: only properties with numerical (or boolean) values will be displayed in fragalysis - this will be reviewed at a later date

Compound Set Download (Optional Extra)

As part of the design process a question was raised on re-creating the “download_structures” functionality for RHS compounds. This appendix sketches out a possible solution to the problems.

Features of download_structures:

1. Select “all” or groups of protein codes to download
2. Select file types in the download (sdf, bound etc) from those available in the target
3. “Preserved” downloads and Latest information.
4. Caching of files for up to an hour to save time building large datasets
5. “extra files” functionality
6. Download file structure based on target upload structure
7. Self-documenting based on the list of structures + a description
8. SDFs either separately or in one file.

There are some differences between targets and compound sets as the compound sets are stored in a very different way:

1. Compound sets consist of a single SDF file containing all the compounds + (optionally) PDBs.
2. The link to the single SDF file is stored, but not individual SDF files (i.e. it’s not split up). Optional PDB files are stored in the media server.
3. Some context supplied in the “blank” molecule is stored in the database.
4. Parameters are stored in the database in a combination of models ScoreDescription and TextScoreValues.

From my quick analysis above, a solution could be constructed around the data in the tables better than from individual files (but with some questions below):

1. Selecting some combination of computed molecule names on the frontend (Would we want to grab molecules from different compound sets?)
2. Selecting an output file type that could be simply constructed from the data in the table (I’m not an expert but maybe a .mol or .sdf) and the parameter information.
3. PDBs attached to the compound set could also be downloaded (so an “include pdbs” option)
4. There would be no possibility of “preserved” downloads unless you went back to the original SDF files that were uploaded - this would be complicated.
5. Caching would be possible - although would it be necessary? - I don’t think we have the same file size issues here?
6. “Extra files” would belong to an individual compound set. If we added this, we’d have to consider where it would be useful to download it.
7. “Self documenting” could be adapted although I wonder if some kind of metadata with all the parameters in the files may be more useful? Do we want these files to be re-uploadable? If so, then we might consider recreating a “blank” molecule at the top.

As a rough ballpark, I would say that the backend changes would be in the order of 3 to 5 days depending on the detailed requirements as I would not be able to reuse the existing functionality. There would be frontend work also required and this would need to be estimated by M2M.