DRAFT: Transitioning VLASS from AAT/PPI to Workspaces

There is a need to retire the old OODT-based workflow system, and in particular the VLASS project's use of the AAT/PPI workflow software as a library for their own set of workflows. This page will document the full extent of the VLASS workflow functionality, and provide a transition plan to using Workspaces instead.

- Existing System's Workflows
  - Require Transition
    - Ready For Transition
    - Investigation and Design Work Required
  - Can Be Ignored & Removed
  - Require Further Consideration
    - calibration ingestion
- Individual Workflow Transition Notes
  - SECI
  - SECC
  - Quicklook
- Transition of Workflow Launching Software
  - Vlass Manager
  - Pimscache
  - Calibration Ingestion
- Other considerations
  - Script and Utility Deployment Location
  - Processing Directory Names
  - Information Granularity
    - Feedback From pimscache split
  - Elimination of External Dependencies
  - Casa Completion Status Clarification
- The Path Forward
  - Potential Structure For Stories
    - Individual Workflows
      - Special Cases:
    - Workflow Launching
      - Pimscache
      - Vlass Manager
      - Calibration Ingestion
    - Internalizing & Updating Utilities
  - Decisions
    - Final List of Workflows
    - Utility Location
    - Definition of Success/Failure for CASA
    - Processing Directory Handling
    - Expanded Visibility within HTCondor for DAG processing
    - Desirability and Level of Feedback from 'pimscache split'
  - Risk and Breakpoints
- Phase 1 Outcomes:
  - Workflows
    - Note The Old Tasks Remaining
  - Pimscache
  - Vlass Manager
- Phase 2 Plans:
  - Workflow Launching Unification
    - Processing Workflows
    - Ingestion/Caching Workflows
      - Remaining AAT/PPI Workflows Used By Vlass:
        - calibration ingestion (triggered)
        - quicklook ingestion (done in bulk after caching)
  - Communications Overhaul
    - Limit Message Collection Scope
    - Persistent Queues
    - Automatic Pimscache Launching

Existing System's Workflows

As of VLASS System 2.3.0, the following workflows are defined:

- calibration
- quicklook
- image-caching
- compression
- se_calibration
- se_cont
- runRestoreWorkflow

- seci_split_measurement_sets
- se_continuum_imaging
    - Workspaces already handling: CASA
- se_coarse_cube_imaging
    - Workspaces already handling: CASA & retrieval of reimaging resources

These fall into several catagories:

Require Transition

The work involved in transitioning individual workflows will be to move the fundamental processing functionality of the workflow to the Workspaces system.   Some preliminary steps will still be performed by the VLASS system.   Those preliminary steps will be transitioned with the changes to the workflow launching systems (see below).

Ready For Transition

- quicklook
- calibration
- se_continuum_imaging
- se_coarse_cube_imaging

Investigation and Design Work Required

- seci_split_measurement_sets
    - There are some existing components of Workspaces workflows that can be reused for this case
    - The management of messaging to the CLI will need to be replicated, if that is desired.
- image-caching
    - There are a great deal of assumptions built into this script about the directory name and structure.
    - This will need to be redesigned, with more information provided by the Manager rather than intuited at run time.
- se_calibration
    - This workflow looks (on the surface) to be functionally identical to calibration
    - Require clarification of the subtleties and determination of whether that can be handled within a single Workspaces workflow

Can Be Ignored & Removed

se_cont

compression

runRestoreWorkflow

Require Further Consideration

These workflows may well be transitioned, but there are discussions to be had surrounding them.   This category will be eliminiated, in time.

calibration ingestion

Vlass makes use of the AAT/PPI calibration ingestion workflow. It is not immediately clear how compatible it is with the counterpart in Workspaces.

Individual Workflow Transition Notes

Some VLASS workflows were created specifically with Workspaces interaction involved, but only to facilitate remote CASA processing. These provide an existing model of funtionality that will be expanded to reduce the footprint of the VLASS workflow server.

SECI

The first instance of the VLASS system performing work through the Workspaces system, in particular to facilitate use of the extra cluster nodes hosted at NMT.   It provided the infrastructure necessary to undertake this project.

SECC

This was a more complicated case, as the use of the Workspaces productfetcher was needed to retrieve some required information for the processing.   A multi-stage workflow was created to facilitate the complication, and that provides the basic idea for the expansion of Workspaces taking over existing VLASS workflows.   This workflow will need to be expanded to include the steps still covered by the VLASS system (PIMS retrieval, automatic QA, and component list generation).

Quicklook

The first full transition of a workflow from VLASS to Workspaces (while simultaneously transitioning Quicklook imaging to using the PIMS system). The project will require the relocation of pimscache (Docker can't see the user directories, so the current pimscache deployment location won't work). This transition also was the first instance of relocating the utilities to the vlass_dev processing area on Lustre.

This process, undertaken as an expansion of the planned transition of Quicklook imaging to make use of PIMS and HTCondor, has exposed a few configuration changes required for the Workspaces system. It has also required an alteration of how Workspaces interacts with HTCondor, due to the necessity of using external python code for the automatic QA. The implications of this will be addressed below.

Transition of Workflow Launching Software

Vlass Manager

This will be a complete replacement of the Manager's workflow launching system, and a significant overhaul of the manager's AMQP message handling.

The workflow launching functionality is largely contained to a single method, which will need to be rewritten in the pattern of the WorkspacesImagingTask: based upon the queue desired, it will prepare, load files to, and initiate the relevant workspaces workflow via REST calls.

Currently the manager obtains information about where an execution is in the process by listening in on messages to and from the workflow server regarding cluster submissions. This will be removed, and replaced with listeners for messages from Workspaces instead. Currently implemented functionality only handles entire-workflow information (initiation, pass, and fail). There is, if desired, the possibility of a Workspaces project which could enhance that granularity, with associated changes to the manger's listening functionality. What will be left of the initial messaging infrastructure will be related to listening for ingestion completion announcements for Execution Blocks and Calibrations (however, see below).

There are also other effects (assumptions made about directory locations, etc) that will have ripple effects from these changes that will need to be investigated and resolved. There are bound to be a few consequences of these changes that will need to be resolved after verification and/or validation.

Pimscache

The changes for this tool will be largely (if not entirely) contained to the 'split' subcommand. The conversion of the workflow initiation will be fairly straight forward. It will essentially be a fresh version of the existing wrapper functionality, but in Python rather than Java. Maintaining the current set of messages back to the command line will require further investigation and word design work.

With the expected removal of the Vlass workflow areas under the vlapipe account, it is worth mentioning that a new location for the tool will be needed. The current expectation is that it will move to the Vlass Lustre working areas, but other locations are possible.

Calibration Ingestion

A special case: Ingestion of vlass calibration is handled via the AAT/PPI's calibration ingestion workflow, mediated by the amygdala system. This process needs to be reconsidered in the light of this transition. it is an additional area of the VLASS manager that will have to change, so is highlighted separately from the bulk of the work.

The compatibility of the Workspaces equivalent is currently under investigation.

Other considerations

The combination of Docker and Condor impose a few constraints along with their greater freedom.

Script and Utility Deployment Location

Both the removal of the workflow deployment locations under the vlapipe account, and the limitations of Docker in regard to what areas are visible has implications for the assumption of VLASS workflows by the Workspaces project.

The pimscache utility will need to be deployed to a new location. It is also probable that the pimscache utility should be absorbed into the Workspaces project when it is time for the workflow management portion of the utility needs to change for this project.

The VLASS project area contains a set of scripts used by the current workflow system, only a portion of which will remain relevant for the Workspaces system, and these scripts are located in an area that the Workspaces Docker system cannot access.

There are a couple obvious choices here:

- Solidify a naming convention between the 3 subareas
    - /lustre/aoc/projects/vlass/
        - vlass.dev.scripts
        - vlass.test.scripts
        - bin & lib vlass.prod.scripts
- Unify with the new pimscache location in the working areas on Lustre **(Preferred Solution)**
    - /lustre/aoc/cluster/pipeline/vlass_[dev|test|prod]/sbin/
    - provides a single location point
    - allows screening so that we only move pieces (like the python QA tools and component list scripts, for instance) required for Workspaces workflows

    - allows a cleaner division between automated-use tools and individual-use tools
        - With pimscache being the outlier in this regard
    - removes old processing detritus from the VLASS project area (test and dev script directories)

The second solution is preferred by Workspaces due to consistency with existing Workspaces-VLASS interactions. These areas already exist, are consistently named between deployment environments, and are already accessible to the Workspaces system (in particular, the workflow service docker container which is the submit host to HTCondor). Utilizing the current VLASS project area will require changes to both the Workspaces system configuration and directory names in the project area. The quicklook full conversion has been prototyped assuming this solution.

Processing Directory Names

HTCondor generates its own processing directory names, and Workspaces has embraced that aspect of the tool. There are two ways to approach the handling of processing directories.

In the case of keeping with Workspaces precedent, the transmission of the working directory name back to the manager is an expansion of the current messaging limitations that will need to be addressed in the design of the message handling changes in the Manager's transition.

Alternately, it could be possible for Workspaces to adapt to providing results in a specified directory. This would reduce the impact of these changes upon the Manager (it can continue to assume paths as before), but require an update to the transitioned workflows in concert with the Manager's transition. This would be a less risky option, and is considered the preferred option.

Information Granularity

What is currently implemented for SECI and SECC is fairly minimal usage of Workspaces, and thus only deals with initiation, success, and failure messages from the Workspaces system. This has been sufficient for the time being, but the loss of intermediate status information due to these transitions in the VLASS Manager is a known change to which the VLASS Ops team would need to adapt.

There might be a possibility of providing some more detail (broad strokes transitions between 'gathering data', 'running casa', 'post casa work' stages of the transitioned workflows), but that would require additional investigation and expansion of Workspaces infrastructure, as well as the message handling changes in the VLASS manager itself. These transitions are not something which is readily available from HTCondor, and thus requires some additional effort (and thus Stories) in order to capture.

Feedback From pimscache split

The current feedback mechanism will need to be reimplemented in the Workspaces system, if it is to be retained. The existing design work on the receiving side (the command line tool) can be at least partially reused for a fresh Python implementation on the sending side (the transitioned workflow itself, or within the Workspaces infrastructure). However that is additional work on top of the basics of performing the fundamental restore-split-cache work of the existing workflow.

Elimination of External Dependencies

The attempt to incorporate automateQA.py and qa.py required some additional setup work and configuration changes. The requirement to execute local code on a local host is contrary to the standard HTCondor methdology of portable utilities for maximum flexibility. While this is something that can be worked around, as demonstrated by the Quicklook transition, it is far from optimal. The current transition process will greatly reduce the number of separate utilities required, but there are still several that could be better codified.

Thus far these utilities have been left flexible, allowing the VLASS Opeprations staff to adjust them as necessary, and only occasionally updated within the version control system. With the transition to a new processing structure under Workspaces, the project would benefit from the solidification of requirements and conversion to a portable structure to better integrate with the updated structures. The set of utilities that are currently foreseen as still required for processing are:

- pimscache
- annotateQa.py (and dependencies)
- annotateQaSE.py (and dependencies)
- create-component-list.sh (and dependencies)
- annotateQaSECube.py (and dependencies)
- create-cube-compoent-list.sh (and dependencies)

The proposal here is that these utilities have their functionality solidified and then are converted to a form better suited to working with Workspace (pimscache already meets this standard). The other utilities would need to be updated to Python 3.8 (minimum), transformed into a portable executable (most likely PEX, similar to pimscache), and having their CAPO functionality expanded (where necessary) to provide flexibility. The new versions of these utliities would then be deployed to the appropriate location (see above) for use in transitioned workflows.

This process has the additional requirement of time spent understanding these tools (in particular the component list utlities) to ensure that these conversions remain true to their function. Some of these utilities (especially those related to coarse cubes, are little more than copies or stubs, and it will take time for them to mature to the point of being ready for conversion. Others, however are far more ready for this process, and have been the subject of requests for SSA time and effort previously. This work can be performed distinct from the transition of the workflow which uses them, providing additional flexibility in the larger project.

Casa Completion Status Clarification

Currently, there are multiple definitions of whether CASA as completed successfully or not across the AAT/PPI, VLASS, and Workflow systems. Examples are gathered below:

- Automatically assume everything is fine
    - Allow any follow-on work to fail instead
- Searching for SEVERE messages from the flagmanager in the log
    - Existence of those messages returns a failure state
- Report CASA's exit status
    - Can result if false-failures
- Others?

This transition to systems using Workspaces is a good time to resolve the multiplicity into a single standard, handled by a Workspaces utility program, allowing for better feedback to users.

The Path Forward

Potential Structure For Stories

Individual Workflows

Each individual workflow is either 1 large story, or two smaller ones (creation of the replacement workflow Workspaces side, and transitioning of the vlass workflow to the new structure, and deletion of replaced code and scripts).   These will get easier the more we do (with Quicklook already underway providing insight that has gone into this document).

There is a limited amount of parallelization that could be done with these stories without causing merging conflicts (especially on the VLASS side of the work, each transition modifies the same set of files in each case).

Special Cases:

As some of the investigation noted above happens, certain workflows may expand beyond the suggested story structure above.

The cases most likely to expand in scope are:

- seci_split_measurement_sets
- image-caching
- compression

But that has not yet been determined. Other factors not anticipated may appear.

Workflow Launching

This assumes that all relevant workflows for each utility have been transitioned. The three subsystems identified in the discussion above provide a beginning for dividing the work, because they are largely disjoint.

Pimscache

Dependent Upon: seci_split_measurement_sets

This case might be best broken down into three separate stages:

1. Transition pimscache to the Workspaces project
2. Transition workflow launching from AAT/PPI to Workspaces
3. Update feedback messaging infrastructure

pimscache is already compatible with Workspaces (being both Python 3.8 and using the pex format) so the first stage might be relatively painless.

Vlass Manager

This is the biggest chunk of work, with the most potential to expand beyond any initial estimates. However, it greatly reduces the number of moving parts in the system. Here it might be best to separate the work for workflow launching and message handling (as seen in the pimscache case), but the largest difference here is that additional time for both verification and for fixes during validation should be considered. Those latter two points are sufficiently emphasized as to warrant consideration as extra stories, to ensure a smooth transition to the revised structure of the project.

Calibration Ingestion

Dependent Upon: determination of the path forward for that functionality, Vlass Manager transition

This project is most simply done after the Vlass Manager has already taken over the role of communicating with Workspaces.   This project is then a matter of reducing or eliminating the special behavior for this particular use case,

There is one additional consideration that has to be handled here (the return of the name of the calibration file to the manager for persistence), which will only slightly complicate matters if it is required.

Internalizing & Updating Utilities

While the story breakdown of these changes seem fairly straight forward, there is a significant risk that that (especially for the first of each kind of utility) that the work could require more effort than initially estimated. In addition, stories for infrastructure changes (repository restructuring, deployment changes, etc) and for developer investigation and familiarization prior to the commencement of the true work would be advisable.   As noted elsewhere, the pimscache utility is largely in the desired state already, and would certainly be a single, small story once the infrastructure work is completed.

It should also be noted that the work to soldify the behavior of the utilities and the determination of the desired set of parameters for potential modification between releases will be prerequisites to any of the above stoies being undertaken.

Decisions

Final List of Workflows

- quicklook
- calibration
- se_continuum_imaging
- se_coarse_cube_imaging
- seci_split_measurement_sets
- image-caching
- se_calibration
    - This can be covered with the same fundamental process as calibration, perhaps with differences in the files provided by the manager

Utility Location

Definition of Success/Failure for CASA

Processing Directory Handling

Expanded Visibility within HTCondor for DAG processing

Desirability and Level of Feedback from 'pimscache split'

Risk and Breakpoints

Each individual workflow transition is self-contained on the Workspaces side, and the VLASS repository changes just require a bit of coordination between developers. After any full transition of a given workflow, this work could pause without hampering the system. Several workflows are ready for transition, and the rest will be ready after a bit more investigation and planning.

The workflow launching infrastructure changes are dependent upon the relevant workflows, but are otherwise largely independent code-wise.   The transition of calibration ingestion management should wait until after the more general changes for the VLASS manager, however. As with the related workflows, each launching platform transition is a point at which the project could pause briefly without hampering functionality.

Phase 1 Outcomes:

After a review of the above notes, it was decided to focus on a smaller goal (transitioning the major functionality of the core Vlass workflows) to use Workspaces.   This left the Vlass project with a significantly smaller reliance on the Torque scheduler.   This necessitates another round of work (see below) to complete the transition off of the Torque scheduler and an overhaul of how the Vlass Manager communicates with other systems.

Workflows

The finalized list of workflow had the bulk of their work converted to being done via Workspaces.   All but one vlass workflow have the exact same layout:

Current Vlass Processing Workflows Defintiion

```
         <workflow name="calibration">
        <task class="edu.nrao.archive.workflow.tasks.ProductLocatorWorkflowStartupTask"/>
        <task class="edu.nrao.archive.workflow.tasks.vlass.WaitForLustreTask"/>
        <task class="edu.nrao.archive.workflow.tasks.GetJobFilesTask"/>
        <task class="edu.nrao.archive.workflow.tasks.vlass.WaitForLustreTask"/>
        <task class="edu.nrao.archive.workflow.tasks.vlass.WorkspaceImagingTask"/>
    </workflow>
```

The odd workflow out consists of the single task (edu.nrao.archive.workflow.tasks.vlass.ImageCachingTask) which uses some of the same tools as the WorkspaceImagingTask.

Note The Old Tasks Remaining

Both preparatory workflow tasks (ProductLocatorWorkflowStartupTask and GetJobFilesTask) are still run via Torque. Their work will need to be redistributed (some to Workspaces, some to the Vlass Manager) in order to entirely free the Vlass processing from using Torque.

Pimscache

The pimscache utility was entirely decoupled from the vlass workflow server (and all torque workflows). Instead, this tool queries the Vlass Manager's database and utilizes the REST endpoints provided by both the Vlass Manager and by Workspaces' Workflow Service.

Vlass Manager

This piece of software was largely undisturbed. Display of stages changed, but that was done primarily with changes to the information persisted to the jobtask database table.

Phase 2 Plans:

This phase completes the transition of the VLASS data processing away from Torque and to exclusively use Workspaces Workflows. It also will address technical debt in an overhaul of the manager's message handling and collect scattered fragments of related code into a cohesive whole.

Workflow Launching Unification

The base work for this has already been done in the vlass workflow, there are helper classes that wrap WS communications and some patterns for handling individual workflow response messages. This work needs to be ported into the manager for use at a higher level.

On a general level, the manager must keep the ID returned by the workspaces creation stored in the database, and it may be beneficial to keep the processing and ingestion WSIDs separate.

Processing Workflows

WS: Take over creation of desired destination directory

Preparatory work has already been done, but should be reorganized. Each of these workflows needs to upload a handful of files, and provide specific parameters. Currently handled in separate methods, these functions might be better handled as separate classes.

Ingestion/Caching Workflows

WS: Handle additional data flow through calibration ingestion for VLASS ('token' in, product name returned). If this functionality is able to be abstracted, that would help with future cachingingestion transitions.

Organize the actions of the 'Accept & Archive' button by queue, and allow easy transition for a queue from one action to another. Each queue will now either launch a caching workflow or an ingestion workflow.

Remaining AAT/PPI Workflows Used By Vlass:

calibration ingestion (triggered)

quicklook ingestion (done in bulk after caching)

Communications Overhaul

Limit Message Collection Scope

The scope of messages from the AAT and Workspaces which interest the Vlass manager are fairly small. The messages of interest are:

1. Status Updates From Workspaces regarding Vlass Workflows
    a. Specifically, listening or the routing key for external messages from Workspaces
    b. Potentially separated by processing vs ingestion. Those two IDs should be kept separately and the results update different state columns in the database.
    c. completion messages for ingestion workflows should lead to the persistence of the science product's name in the product_version_archiveids table
        i. Calibration currently requires a format of: {'NAME':#} in this table
            1. the number is not relevant, can be replaced with the workspaces workflow id
            2. Bonus Points: update both the persistence mechanism and pimscache to handle a simpler formatting with just the product name.
2. Notification of observations being ready for calibration from mr_books.
    a. Limit messages to 'ingestion-complete.observation' rather than all types.

All the message handling functions should gracefully (but not necessarily quietly) exit if they encounter missing information or unexpected data.

Persistent Queues

Change all AMQP queues (there will only be 2 or 3 total) to be persistent, with clear names regarding their function. This will facilitate catching up on work from any downtime from the system, requiring less manual intervention.

Automatic Pimscache Launching

Revisit the mechanics of automatic pimscache execution in the new system. Do we still require the additional sudo permission for tomcat to execute pimscache? If not, remove it to make CIS/SIS happier with us.