UNIVERSITY OF HAWAII AT MĀNOA
Institute for Astrononmy

Pan-STARRS Project Management System

IPPTools Software Design Description

The IPP Analysis Stages: Job Relationships, Database Tables, and Data Flow

Grant Award No.	**: F29601-02-1-0268**
Prepared For	**: IPP**
Prepared By	**: Eugene Magnier**

Document No.	**: PSDC-430-019**
Document Date	**: June 15, 2006**
Revision	**: DR**

DISTRIBUTION STATEMENT
Approved for Public Release – Distribution is Unlimited

Submitted By:

[Insert Signature Block of Authorized Developer Representative] Date

Approved By:

[Insert Signature Block of Customer Developer Representative] Date

Contents

1 Overview

This document defines the tasks used by PanTasks to control the data analysis stages of the IPP. The document examines the specific tasks, the related Metadata Database tables, and the pipeline management commands used to connect these components together. The Metadata Database is used to store the current IPP analysis state, as well as result processing data points. The tasks needed to define each of the IPP analysis stages (Phase 1-4, detrend creation, etc) are illustrated in this document and the relevant MDDB tables are listed. The collection of diagrams shows the IPP tasks and the Metadata Database tables needed to manage the flow of data through the system. This document does not discuss in depth the interactions of various analysis programs with either DVO (the IPP Astrometric and Photometric Object database) or Nebulous (the IPP large data file management tool).

2 PanTasks Summary

PanTasks is the IPP tool which manages the sequencing of data analysis steps and, with the related tool 'PControl', distributes the data processing across a cluster of computers. This document is not a user's guide to PanTasks. We will briefly discuss the capabilities of PanTasks to give the reader a basic working knowledge; for further usage details, please see the PanTasks user's guide.

The purpose of PanTasks is to manage the automatic construction and execution of inter-related (often repetative) operations. PanTasks uses a set of rules to define UNIX commands, and their corresponding command-line arguments, to be performed on some regular, repeated basis. The utility of PanTasks is that it can easily define an analysis system which is completely state-based, as opposed to an event-driven system.

The two basic units of PanTasks operation are the 'task' and the 'job'. A 'job' is simply a command the user would execute on a command line; it consists of a command along with optional command line arguments. A 'task' is a generic description of a type of job in which the details of any specific piece of data are omitted. The task defines the UNIX command which corresponds to the job, and it provides rules for determining the identity of data for the job. The task also defines tests which are used to decide if the job may be executed. Finally, it defines a polling frequency in which PanTasks should attempt to construct a new job for the task.

For example, we may want to regularly copy files from one location to another. Perhaps the name of the next available file is available in a database table, and can be retrieved with the command 'nextFile'. Perhaps we wish to check for new files every 60 seconds. Thus, the task is to copy files, while a specific job of this task might be of the form `cp newfile newpath`. With PanTasks, we would define a copy task somewhat like the following:

```
task copyfile
  periods -exec 60.0

  task.exec
    $file = 'nextFile'
    if ($file == "none")
      break
    end
    command cp $file $newpath
  end

  task.exit 0
    queuepush copied $file
  end

  task.exit 1
    queuepush failure $file
```

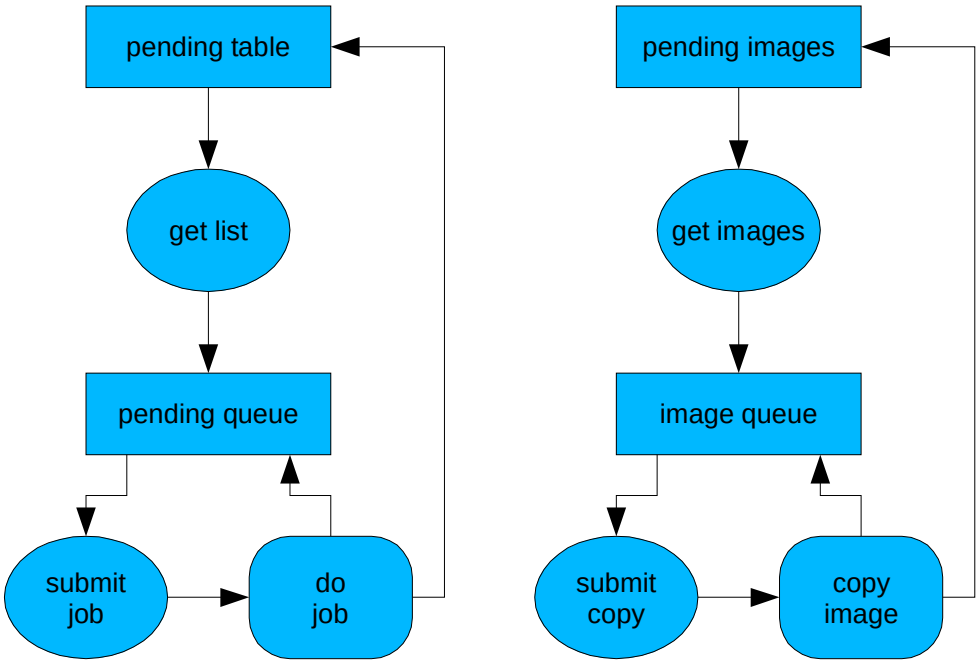

Figure 1: PanTasks queues and MDDB tables

```
    end
end
```

In this simple example, the task is attempted every 60 seconds. If there is no new file (output of 'nextFile' is 'none'), then no job results from this task. If there is a new file, the copy command is performed. This example is deceptively simple because one could easily imagine writing a stand-along program which performs the same thing. The important advantages of using PanTasks for this type of operation are:

- the output from one job can be used to spawn a number of other tasks.
- the success or failure state of each job can be used to spawn other tasks.
- the details of the job and data test are kept separated from the rules which connect tasks together.
- the relationships between tasks are kept together in a single location.

In addition to these organizational advantages, PanTasks also provides a direct connection to the tool for monitoring parallel jobs, pcontrol. Thus the jobs spawned by PanTasks can be defined to run in the background locally or on any of the computers in the parallel processing cluster.

3 Persistent vs Ephemeral State in PanTasks

The IPP, a fairly complex analysis system, uses PanTasks to select jobs, distribute them to the cluster, and harvest the results. It uses the Metadata Database to record the results of a given analysis step, and to determine which jobs must be performed when.

There are some subtleties in the interaction between PanTasks, the Metadata Database tables which store the system state, and the jobs which are currently being performed. There is a choice to be made between rigorously maintaining the system state in the Metadata DB at all times or keeping an intermediate set of state tables. Keeping the exact system state in the Metadata DB tables would require many extra queries to/from the database and may introduce additional latencies which are undesirable. This is because any attempt by PanTasks to initiate a new job would require PanTasks to mark the corresponding data item in the Metadata DB (the item which acts as the trigger) with a 'pending' state, and then mark it again as 'done' when the job actually completes. This also has the drawback that, if the system crashes (eg, hardware failure), some initial process would be required on start up to find all Metadata DB items which are in the 'pending' state (examining all possible items which can be in such a state) and reset them to the 'new' state.

We implement an alternative in which PanTasks maintains an internal, ephemeral stack of the pending jobs, and only updates the system state entries in the Metadata DB when jobs are actually completed. In this scenario, as far as the Metadata DB tables are concerned, data items transition only between a 'new' and a 'done' state. Any jobs which are pending when the system crashes or the power is lost are simply dropped, and will be automatically re-constructed when the system restarts. In this paradigm, no intermediate operation state is saved, and no partially completed job can be recovered. Since the IPP is defined in terms of a fine granularity, with jobs lasting no more than 30 - 120 seconds, crashes under this model will not have a large impact on the data processing.

Figure 1 illustrates this ephemeral vs persistent state information and the interrelation between the metadata tables and PanTasks. The left-hand portion of the diagram illustrates the recommended interaction between the metadata database tables and PanTasks' internal queues. Some table in the metadata database defines a list of data items which are to be processed by some analysis job. PanTasks uses a two-step approach to define the analysis jobs based on this list. First, one task queries the MDDB for a list of pending items, adds the returned items to an internal PanTasks queue. The process of adding the elements to the queue is defined so that only unique items are added: already existing items are skipped. The entries in the queue consist of the data items of interest and an internal temporary state. At first, this would be 'pending'. A second tasks pops 'pending' entries one-by-one from this internal queue, submits a job based on the entry, and sets the temporary state in the internal queue to 'running'. The internal state is needed to prevent PanTasks from re-submitting a job for the same data item before the first job is done or assessed. Since the job make take an arbitrary amount of time, PanTasks requires a mechanism to remember which data items it has already submitted. When the job eventually completes, the metadata database table is updated noting the completion. This may be done either by the job itself or by PanTasks as part of the job exit rules. In addition, the state of the entry in the queue can be set to either 'done' or the entry can be simply removed from the queue.

The purpose of this interaction is to maintain the temporary state information within non-persistent elements of PanTasks rather than using the metadata database tables to store this information. This concept has two advantages. First, PanTasks internal queues are in memory and relatively small, thus interfacing with them is quite fast for PanTasks – this should reduce the system latency. Second, by keeping this information non-persistent, the system responds correctly to stopping and restarting PanTasks. Any jobs which have not been completed will not be marked in the database, and will be restarted naturally by PanTasks. The alternative, of writing a temporary state marker in the database would require PanTasks, on startup, to initially clean all database tables of these temporary state markers.

The right-hand portion of the diagram illustrates this process using the process of copying the images from the summit as an example. The metadata database table of interest in this case is the list of pending images, with entries supplied by a job which queries the summit data systems. The job which is actually performed is a remote copy of the image file from the location specified by the summit data system to the appropriate location within the IPP Image Server (Nebulous). (As an alternative to the above, the 'pending images' table may be part of the summit database system, and the 'get images' command may query the summit directly. In this scenario, the 'copy image' command reports to the summit data system that an individual image file has been copied.)

In the rest of this document, the use of PanTasks internal queues to manage the temporary data states is glossed over and

assumed part of the tasks defined in the process.

4 IPP Pipelines Overview

The IPP as a whole performs all of the image analysis functions required by the Pan-STARRS telescopes, including images from the full Gigapixel camera (or cameras), the test camera TC-3, and the SkyProbe camera. The IPP is designed to be very flexible, with instrument specific details isolated in configuration files associated with the different cameras known to the system. As a result, the organization of the top level analysis infrastructure must be sufficiently general that a wide range of cameras can be accomodated. We have a few general principles regarding constraints on the data to be processed which are used to guide the IPP design and developement:

- **Camera Focal Plane Hierarchy** The IPP analysis programs assume that the images to be processed are obtained by a camera which can be represented by our Camera Focal-Plane Hierarchy of data structures. This hierarchy is discussed in detail in the Modules SDRS, and defines a top-level *Focal-Plane Array (FPA)*, which may contain 1 or more *Chips*, each of which may contain one or more *Cells*. An *FPA* is identified as having a single optical system feeding photons to the detectors. A *Chip* is identified as a unit of data all deriving from a single detector (piece of silicon), while a *Cell* is identified as a collection of pixels read out as a continuous cartesian grid. Finally, a single collection of data from an *FPA* may include multiple *Readouts* from any or all of the *Cells*.
- **Exposures vs Groups** The processing presumes that the data is organized into *exposures* and exposure *groups*. An exposure represents the data from a single FPA, with the possible subdivision of the exposure into multiple readouts for some or all of the cells. Exposure *Groups* are any group of exposures which are related together in some way; the definition of the *Groups* may be provided by the observers, or they may be derived from the characteristics of the exposures. The use of a particular *group* depends on the context of that group. A few examples of exposure groups:
  - a dithered sequence of exposures to be stacked for cosmetics and improved signal-to-noise.
  - a twilight flat-field sequence.
  - all images of the same filter within a 10 degree region to be used to construct an sample astrometric reference.
- **Image Files (imfiles) vs Exposures** Any single exposure may consist of a number of different data files. The number of *imfiles* for a given exposure will depend on the camera, as will the data organization within those image files. Also, a particular camera will supply files corresponding to one of the particular Focal-Plane Hierarchy elements. The IPP analysis must be able to interpret the incoming data correctly.

As discussed elsewhere, there are several major types of analysis performed by the IPP. For the purposes of data organization and parallel processing efficiencies, we have identified the following divisions of the analysis tasks. These will be discuss in much more detail below.

- **Science Image Analysis** : This represents the analysis performed on the images obtained by the telescope, and generally performed in real-time, night-by-night. The science image analysis tasks are further subdivided as follows:
  - **Phase 1** : The full focal-plane array is examined quickly to determine an initial astrometric calibration. In this step, the OTA guide stars may be used as the astrometric reference; if none are available, predicted bright star positions are examined. This step is only used for mosaic images, and may be skipped if no guide stars are available *and* the astrometric calibration for the telescope / camera is reliable (better than 10 arcseconds).

  - **Phase 2** : Each image file is analysed independently: the image is detrended (bias, dark, flat, fringe, etc), sources are then detected to a modest level, improved astrometric calibration is performed.
  - **Phase 3** : The collection of sources measured from all of the image files for the camera are used to determine a global astrometric, and possibly photometric, solution for the exposure. This step is only required for mosaic cameras.
  - **Phase 4.1** : An exposure group consisting of images obtained in a specific region of the sky are merged together. In this step, the images are first warped to a common pixel grid, defined by the static sky images. The collection of images are then used to construct a single, cleaned image by rejecting the outliers from the source images in the stack. The corresponding static sky pixels are then used to construct a difference image from the resulting stack.
  - **Magic** : In this step, the difference images are examined to find the trailed images introduced by artificial satelites. These so-called *streaks* are excised from the difference images, as well as all of the source images which were used to generate the difference images; the public data sources are updated with the precise, correct time. Note that this step requires that separate difference images be generated for each of the input images, a step which would be skipped if *magic* were avoided. Also note that, until *magic* is performed, the publically available time has a limited precision (probably $\sim$ 1 minute errors). This step is only necessary in the operational IPP system given the restrictions from the Air Force.
  - **Phase 4.2** : After *magic* the final difference and the final cleaned stacked image are produced and objects in both images are detected. The difference sources are used to mask the extreme outliers in the cleaned stack, which is then used to update the Static Sky images.

- **Static Sky Image Analysis** : While the science image analysis is performed as images are availablef, the static sky image analysi occurs on a very different timescale. In steady state, the full static sky analysis will take place over the course of a full year. At any given time, the portion of the sky corresponding to the location of the sun will be under-going the analysis. In practice, for PS-1, the static sky is produced in a somewhat different fashion than in the steady-state model. In PS-1, the different survey strategies introduce very different update rates for the static sky. At one extreme, the AP Survey will not have enough data for a complete static sky analysis until nearly 22 months after the survey begins. At the other extreme, the deep survey, which observes a much smaller portion of the sky, may best be analysed quite frequently. These details are part of the science guidelines of the PS-1 surveys, and are beyond the scope of this document. Rather, the IPP Static Sky Image Analysis must provide the capability of defining the static sky analysis in a flexible and dynamic fashion.

- **Basic Detrend Creation Analysis** : The analysis of most of the detrend data is grouped together in a common analysis stage. The differences between the analysis of the bias, dark, flat, and fringe images is primarily one of how the input images are pre-processed, what statistic is used to characterize a given input image, how the input images are scaled before being combined, and what normalization is applied to the resulting image. All of these types of detrend images can thus be processed with a single analysis pipeline which is made aware of these minor differences. This stage is never the less fairly complex, and as a result is subdivided into several compenents, as discussed below.

- **Other analyses** There are a number of other tasks which the IPP must perform that are not well-defined by the different analysis types discussed above. Some analysis tasks are not automatically triggered, and are thus outside the scope of this document; these are the tasks which are more properly considered as research projects than analysis systems. The other important automatic tasks are:

  - **Summit Copy** : In this stage, the data source or data sources are queried for new exposures and image files, which are then copied to the IPP data area. This stage also includes the copying of other metadata which are not included in the image files.

- **Image Classification** : new images which are introduced to the IPP are examined by this analysis stage and placed in the appropriate table for processing. This step includes a small amount of accumulating statistics about the images.
- **Data File management** : a few tasks are necessary to monitor and maintain the clustered storage system. These tasks include the automatic duplication and deletion of different types of files from Nebulous, the file storage archive. This also includes automatic redistribution of machine assignments as hardware is added or removed from the system. This collection of tasks also includes monitoring of system parameters to alert people in case of dangerous hardware situations.
- **Irregular Calibration Data** certain types of calibration information is extracted on different intervals from the more regular detrend images. These types of calibration data include improved telescope pointing models, astrometric calibrations, photometric calibrations, flat-field correction frames.

5 Tables, Tasks and Tools

The following sections discuss the database tables, the tasks within PanTasks, and the collection of programs used by PanTasks to examine and manipulate the state tables. These later programs do not, in general, perform any in depth analysis; instead they perform actions such as selecting from one table images ready for analysis in a following processing step. This collection of tools is grouped under the name of the `ippTools`, and consists of a separate tool for each of the different major analysis steps.

The `ippTools` make use of *glueforge* to simplify the management of the database table schema. Glueforge provides a single mechanism to generate a collection of C data structures, database tables, database access APIs, and I/O routines from a simple table description configuration file. All APIs generated by *glueforge* for the same type of interaction have common naming schemes. This technique has several important advantages. It makes the writing of C database interactions very quick and easy. It also makes it easy to modify the database schema without disrupting the software development. Finally, it provides a simple, self-documenting source for data structure of multiple types which can be shared between programs or platforms.

Within the following diagrams, we illustrate the database tables used to track the state of the IPP. We also show the commands provided by `ippTools` to connect the tables. Finally, we show the IPP tasks which initiate the different analysis steps. The following set of diagrams uses several consistent features. The blue-and-grey rectangles define the metadata database tables. The blue section contains the table name, while the grey section lists a minimal subset of the table columns. The ellipses represent programs (or program portions in some cases) executed by PanTasks. The blue filled ellipses represent the `ippTools` commands which are executed locally on the computer hosting PanTasks. The grey-blue ellipses represent the commands executed on the parallel cluster, monitored by `pcontrol`. The green ellipses represent commands executed by hand for testing and manual intervention.

In most of the analysis tasks, we use a two-table approach to the data in order to avoid excessive latencies. One table is used to track quantities which are still pending for a particular stage. When the analysis is completed, these items are moved from the 'pending' tables to corresponding 'done' tables. Although this introduces a somewhat higher number of tables and complexity, it will avoid the system from slowing down as the number of data items grows with time. The pending tables are searched repeatedly by the `ippTools` programs as they attempt to select new data of interest. In contrast, the done tables are searched much less frequently.

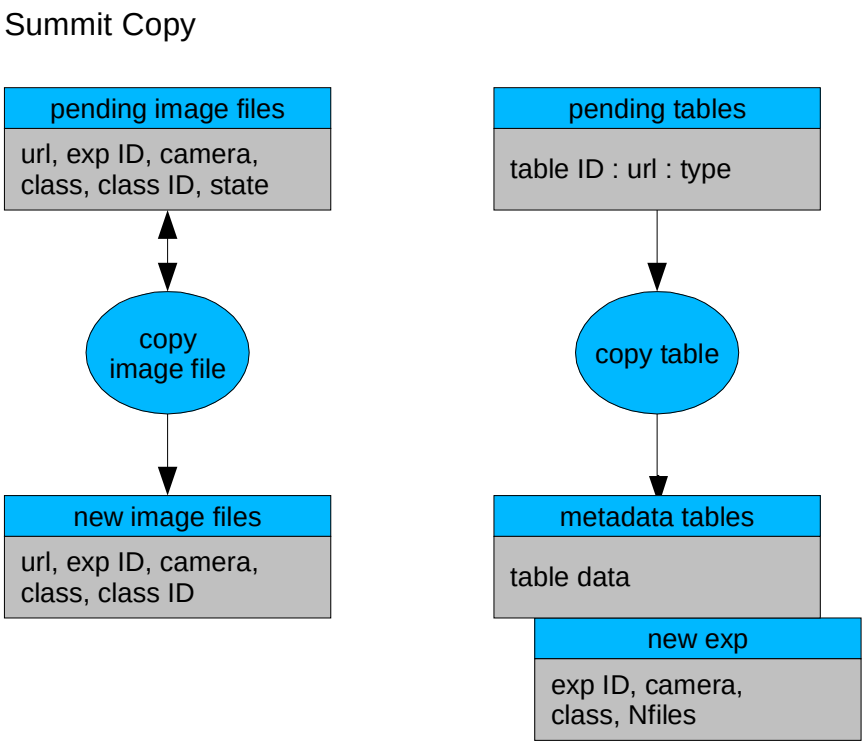

Figure 2: Summit Copy Tasks

6 Summit Copy Tasks

Figure 2 illustrates the MDDB tables used to copy data (images and metadata tables) from the summit. The left-hand portion of the diagram shows the tables involved in copying images from the summit system. The table of pending image files lists the URLs of the individual image files available for transfer, along with their associated exposure ID and the camera which generated the image. Two other entries assist in interpreting the file: the class and the class ID. The final entry in this table is the current copy state of the file, can have the value of ‘ready’ or ‘copied’.

The class defines the data grouping represented by this image file and may have values of: FPA, Chip, Cell. This value indicates that the provided image file represents the specified portion of the camera FPA. If the value is FPA, the file represents data from a complete FPA, though the file may contain pixel data in multiple extensions or other groupings to be identified later. If the value is chip, the file contains only data for a single chip, presumably of multiple chips available, and equivalently for Cell. Further discussion of the FPA image hierarchy is given in the IPP documents (eg, Modules SDRS). The class ID gives the identifier used to name the class level corresponding to this file. This value is necessary to make decisions on how to copy the data based on the chip / cell before the data is available to IPP components. Table 1 lists likely values for the class and class ID for some common cameras. The system described is sufficiently flexible to allow us to transfer the GPC images by cell if we eventually decide that is more efficient.

The copy process copies the file from the given URL to the appropriate IPP node and adds an entry to the table of new image files, consisting of the same information as the pending image file table, though with a new value for the URL. This URL may be an explicit filename, a reference to an entry in the image server, or a web address, or located on the image server (marked with file:, neb:, and http:, respectively). (TBD: other possible file storage types? perhaps the path could be abstracted without going to the level of the image server? eg: ref:DIR0001/file0001.fits might be in a directory which is defined in a table of directories.) After an image file is successfully copied, the corresponding state in the ‘pending chip’

Table 1: Camera and Data Classes

camera	class	classID
GPC	chip	chip02
skyprobe	fpa	sp01
Megacam	fpa	MegacamSpliced
Suprime	chip	chip0

table is updated from ‘ready’ to ‘copied’.

The right hand portion of this diagram illustrates the process of copying a metadata table. The table of pending tables lists the URLs for the tables which are ready, a unique table ID for each table, and the table type. The copy function copies the listed table and uploads the data to the IPP version of the same metadata database. Two examples of metadata tables needed by the IPP for the basic image processing system are illustrated: the table of new exposures and the table of pending matches. The first lists the exposures which are avilable from the summit system, and all represent entries which are available from the Image server. the second represents the matches between exposure IDs and chips

7 Phase 0

Figure 3 illustrates phase 0, in which the image files are categorised, examined for summary information and basic statistics, and moved to the later phase ’pending’ tables to trigger further analysis. The command `p0search -pending` examines the ‘new imfiles’ and ’new exposure’ tables. It selects images from this table which have not yet been examined (state is ‘new’). These are returned to PanTasks, which sends each image file to a separate analysis node running the `p0search -update` command. With this command, the file header is examined and relevant metadata is extracted (eg, RA, DEC, times, and so forth to be defined later). The process may also select a portion of the image pixel data to determine a rough bias and background level. These statistics, whether derived from the header or the pixel values, are placed along with image summary information in the ‘raw image files’ table, and the state field of the ‘new image files’ table is set to ‘ready’.

The `p0search -update` command is also responsible for moving the exposures to the tables used for triggering the analysis process. If the image class is FPA, the image can be advanced without waiting for any other image files. If the class is Chip or Cell, the process must also examine the ‘new exposure’ table for this exposure ID. The number of class files available for this exposure is listed in this table. The process must the select all image files matching the exposure ID with state of ‘ready’ and compare the number avalable to the number expected. If the two match, then a new exposure is ready. Based on the image type (from the most recently examined image file header or new exp table?), the exposure is added to the ‘raw exposure’ table for images of that type. The allowed types are ‘detrend’, (all bias, dark, flat images), ‘object’, ‘focus’(??), etc. (** The different tables represent different analysis modes. This process also adds an entry to the exp ID / image file match **). This process also adds all science (OBJECT) exposures to the P1 exposure table (for mosaic data) or the P2 chip table (for single detector data). These tables are used to trigger the Phase 1 and Phase 2 analysis stages.

8 Phase 1

Figure 4 shows the tables involved in running the Phase 1 analysis stage. There are paths for exposures to enter the analysis automatically from the Phase 0 analysis (arrow on left) or to be added manually based on a selection from the raw exposure

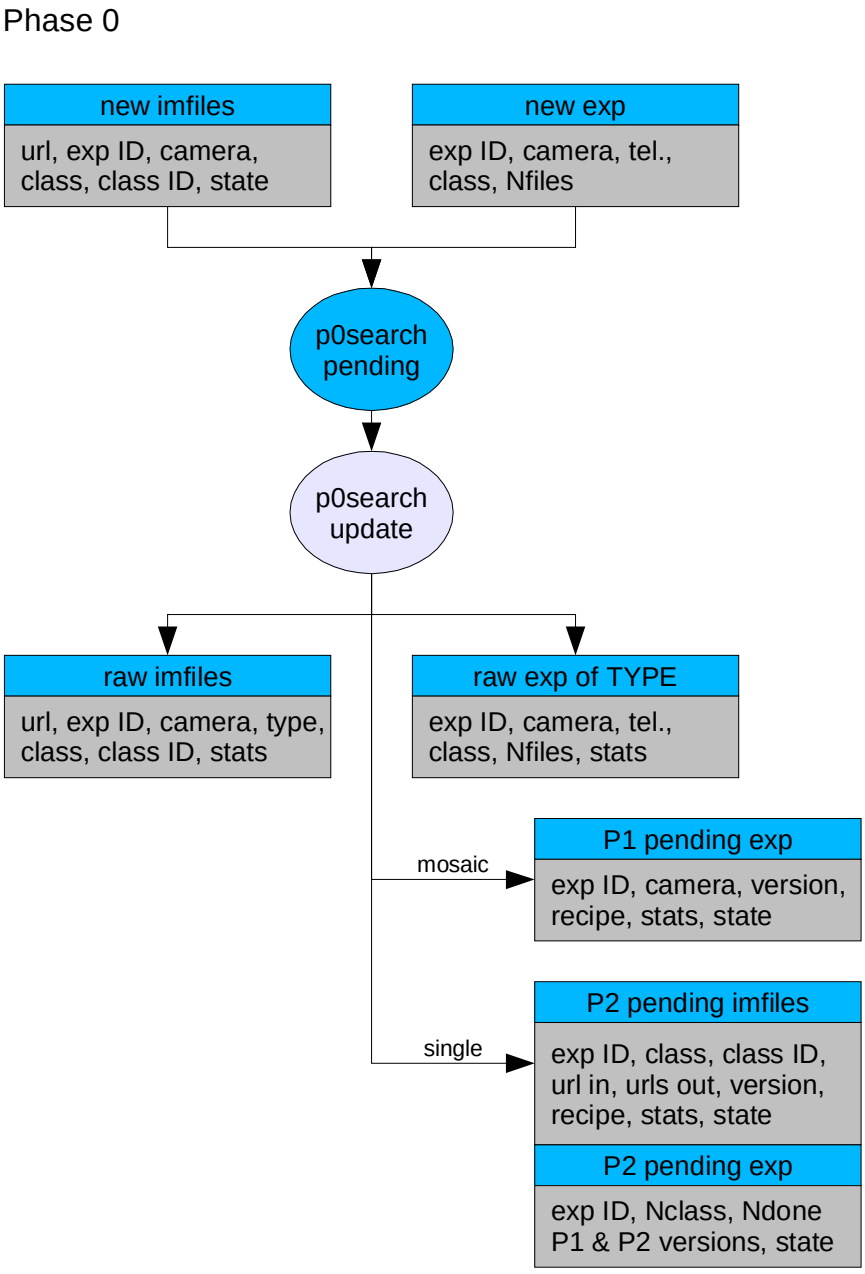

Figure 3: Phase 0 Tasks

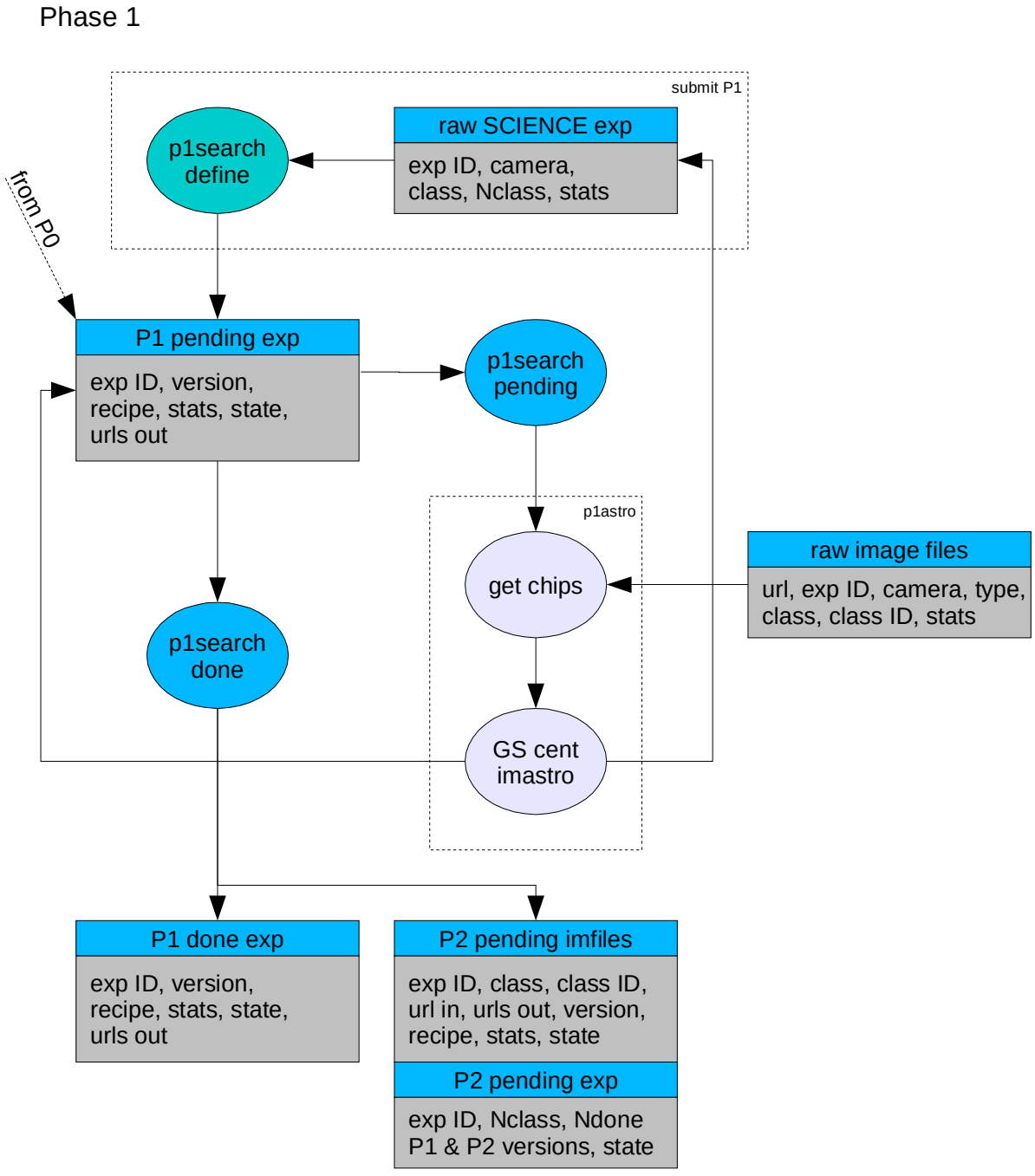

Figure 4: Phase 1 Tasks

table. Exposures to be analysed by Phase 1 are added to the P1 exposure table with the state ‘new’. Exposures may be added multiple times for processing and reprocessing. The P1 done exposure table keeps a record of the old attempts for debugging and analysis. Each time an exposure is added to the P1 exp table, it is given a new, unique version number, allowing the system as a whole to track different analysis attempts. This method is used in all of the image analysis stages (and extrapolated to iterations in the detrend analysis steps below). The top portion of the diagram shows the use of the command `p1search -define` to select and submit an exposure or a group of exposures, potentially selected on the basis of a query from the raw science exposure table.

The P1 pending exposure table is examined by `p1search -pending` to select the new exposures, which are sent to PanTasks. PanTasks initiates a separate analysis job (p1astro) for each exposure, which are sent to the parallel processing nodes. Within the analysis job, the chips (image files) associated with the exposure are select from the raw image file table. The analysis examines the contents of these files, either extract the guide star information from the image files (GS table extension) or searches for and centroids the pixels on appropriate bright stars. The analysis results in astrometric calibration terms which are written to the astrometric calibration file for this exposure. The location of the astrometric calibration file and the statistics of the measurement are written back to the P1 exposure table. The images associated with exposures which are successfully processed by P1 are then added to the P2 image table, which is used to trigger the Phase 2 analysis. This last step is performed by the command `p1search -done`, which is executed regularly to search for completed Phase 1 jobs.

9 Phase 2

Figure 5 shows the tables involved in running the P2 analysis stage. There are paths for images to enter the analysis automatically from the P1 analysis (arrow on left) or to be added manually based on a selection from the raw exposure and raw image file tables. Image files to be analysed by Phase 2 are added to the P2 pending imfiles table with the state ‘new’. When images are added to this table, a single entry is also added to the P2 exposure table listing the P1 and P2 versions for this exposure. These version numbers must be integers starting with 1. If this image did not have a P1 analysis, the P1 version is set to 0. Exposures may be added multiple times for processing and reprocessing. The P2 image table keeps a record of the old attempts for debugging and analysis. As with P1, each time a collection of associated images from an exposure is added to the P2 image table, it is given a new, unique version number, allowing the system as a whole to track different analysis attempts. Note that these version numbers are unique for each *exposure* processed by Phase 2, not just for any image file. The top portion of the diagram illustrates the behavior of the commands `p2search -define` and `p2search -quick`. The first may be used to re-submit the images for an exposure or a group of exposures, potentially selected on the basis of a query from the raw science exposure and raw image file tables. The second version sends images files directly to PanTasks for processing; these entries will not be included in the processing tables, and is used only for testing purposes.

The P2 pending image table is examined with the command `p2search -pending` to select the ‘new’ images. These images are used by PanTasks to generate P2 analysis jobs, running the analysis command `ppImage`. The P2 analysis uses the input url to find and load the image file. The url may be a file on disk, an entry in the image server, Nebulous, etc. The master detrend images matching the specific science image and the conditions are selected by examining the table of master detrend frames. The specific detrend image files are selected by using the master detrend ID to select the matching the entries in the table of master detrend files. After the analysis, the output image, mask, and FITS table of objects, including the astrometry calibration, are written back to the P2 image table, along with summary statistics from the P2 analysis. The state is also updated (to ‘done’).

The completed images are examined by the command `p2search -done`, and when all image files for a single exposure are completed, this command migrates them to the P2 done table. This process is also responsible for populating the P3 pending tables so exposures may be processing by Phase 3.

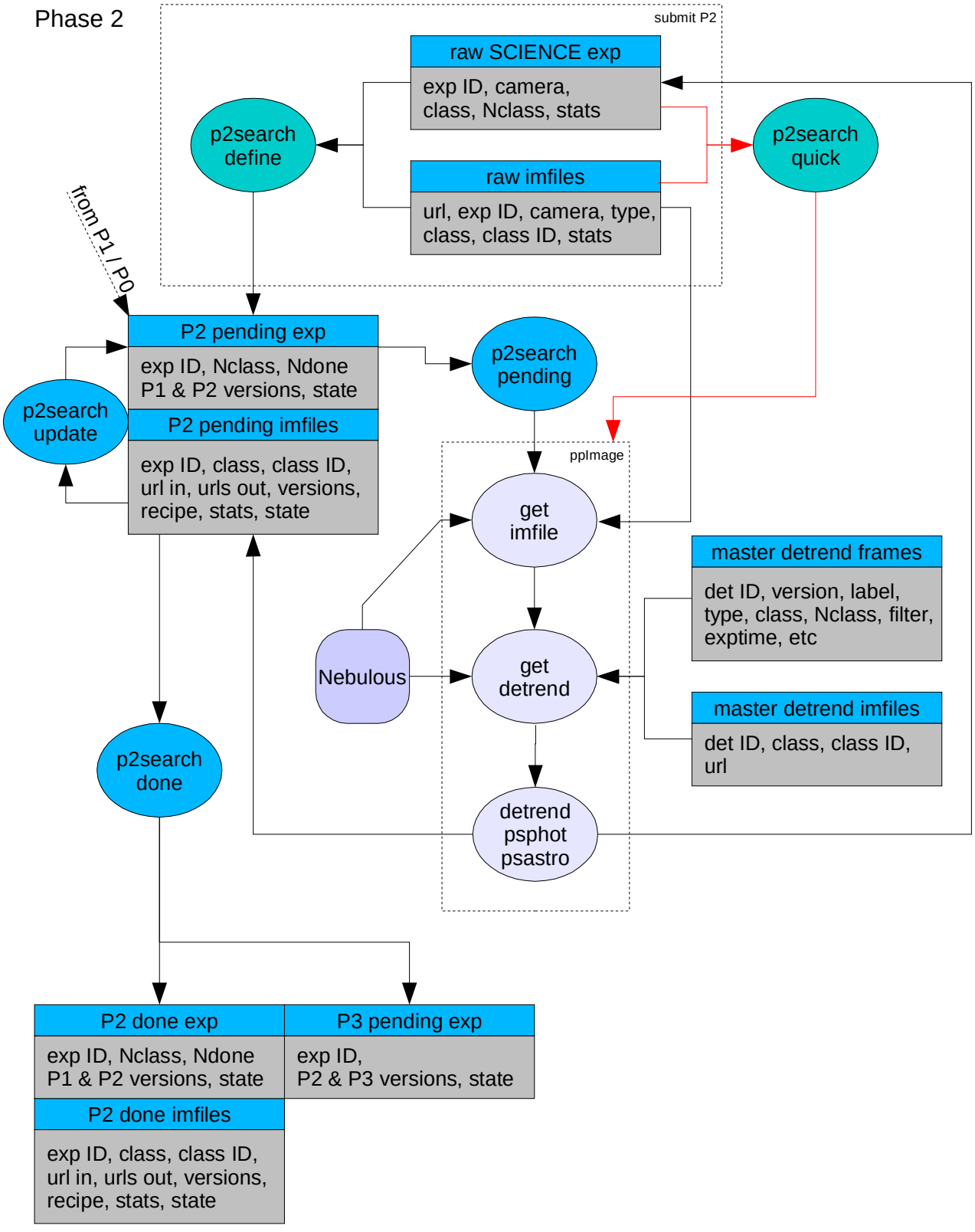

Figure 5: Phase 2 Tasks

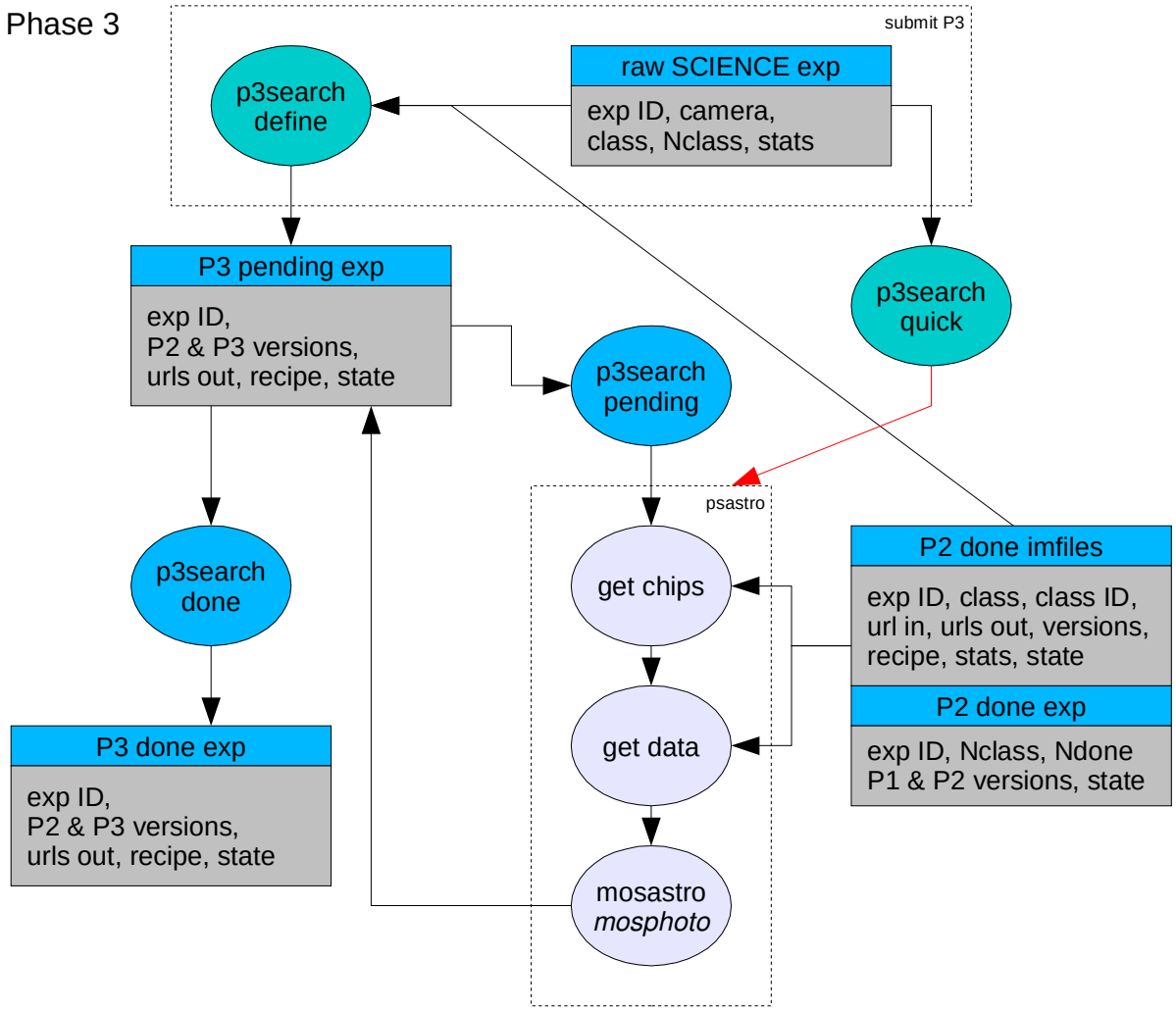

Figure 6: Phase 3 Tasks

10 Phase 3

Figure 6 illustrates the tables and commands involved in the Phase 3 analysis. The P3 pending exposure table lists the exposure ID, the P3 analysis version, the P2 analysis version to be used as input to this P3 analysis, and the recipe to be used. The command `p3search -pending` extracts exposures from this table and provides them to PanTasks for processing. PanTasks launches a Phase 3 analysis (the command `psastro`?) for each exposure. In this analysis, the P2 exposure and image tables are used, in conjunction with the P2 version information, to select the P2 output measured objects and the astrometric calibrations from P2 and P1. These measured objects are matched with the reference catalog objects, and calibrated astrometry *and eventually photometry* is produced for the full exposure. The location of the resulting astometry calibration table is stored back in the P3 exposure table. If the recipe file specifies, the 2-D photometric and background / fringe corrections may also be performed at this stage. Since these analyses require reference data, the recipe may also be used to skip these analysis if such reference data is unavailable or unreliable. At the end of Phase 3, the objects from the exposure are inserted into the photometry database (this is not shown).

The astrometric calibration portion of Phase 3 is principally needed for a mosaic camera. For single-chip cameras, Phase

3 may be used to perform the photometric calibration and simply pass the astrometric results along to the output file to be listed in the P3 exposure table. In this way, later stages of the analysis (ie, Phase 4) can use the P3 exposure table as input for all cameras, even if all the funcionality of Phase 3 is not necessary for that camera. This would be the case for the skyprobe camera, for example.

11 Phase 4

At the end of Phase 3, the images are ready for Phase 4. The Phase 3 diagram shows the output line adding the exposures to be processed by Phase 4 to a Phase 4 table. However, this line is just for illustration purposes. The rules for initiating a Phase 4 run are somewhat more complicated than those for running Phases 1-3. Groups of exposures which have an appropriate overlap should be chosen for the Phase 4 analysis. In the steady-state period of PS-4, it may be straightforward to choose the exposure groups: they would simply be the exposures obtained nearly simultaneously by the four separate cameras. The circumstance for PS-1 will be much more complicated (and even PS-4 will probably be more complex than it seems at first glance). For example, in PS-1, we will not have a static sky for most of the AP Survery. In this circumstance, we cannot run P4, at least until after the complete AP Reference catalog is built, and potentially all exposures re-run through Phase 3. It may be useful for the AP Survey data to split the Phase 4 analysis into two stages: image combination and image differencing. It may even be the case that only the combination portion of Phase 4 is performed on the AP Survey data.

More generally, the image groups selected for Phase 4 analysis may be chosen on the basis of a query of the AP Database (DVO) with some rules.

Phase 4 run can be defined by selecting an observation group, a set of exposures, or a set of rules related to a spatial region (eg, region, time range, and filter.

Phase 4 discussion (and diagram) needs more work

12 Analysis Version and Recipes

Note that each of the stages P1-P4 refer to the processing version from the previous stage. This allows the processing stage to request the correct version of the results from the previous stage, and makes it possible to run and re-run the analysis at any stage without deleting the earlier results. As different analysis attempts are performed for a given image, the versions branch out.

Also note that at every stage, the entries include a recipe identifier. This is used to select the analysis recipe which should be used for this version. By default, the recipe should be set to the current best recipe (use a default name for this?). This feature allows the user to run test analyses with variations on the recipe without altering the analysis system. For example, it is possible to use a different flat-field set by specifying alternate rules for the flat-field selection in a recipe file. If it is necessary to run the P1-P3 analysis with the raw master flats, for example, the user simply defines that selection in the recipe file and submits the images of interest to P1 (or P2, etc), with the corresponding entry for the recipe.

The recipe file may also be used to specify alternative analysis paths and desitinations. For example, it is not necessary that all analysis stops with P4: the recipe file may be used to halt the analysis at P2 or P3. In addition, the recipe file may be used to specify an alternative destination for the output results. For example, to generate the photometric flat-field correction frame from a collection of dithered images, the user may not want the photometry results in the main DVO database. By using the recipe to set an alternative DVO database target, and by specifying the use of the raw master flat rather than the corrected one, the analysis of the dithered images is kept isolated from the other photometry database

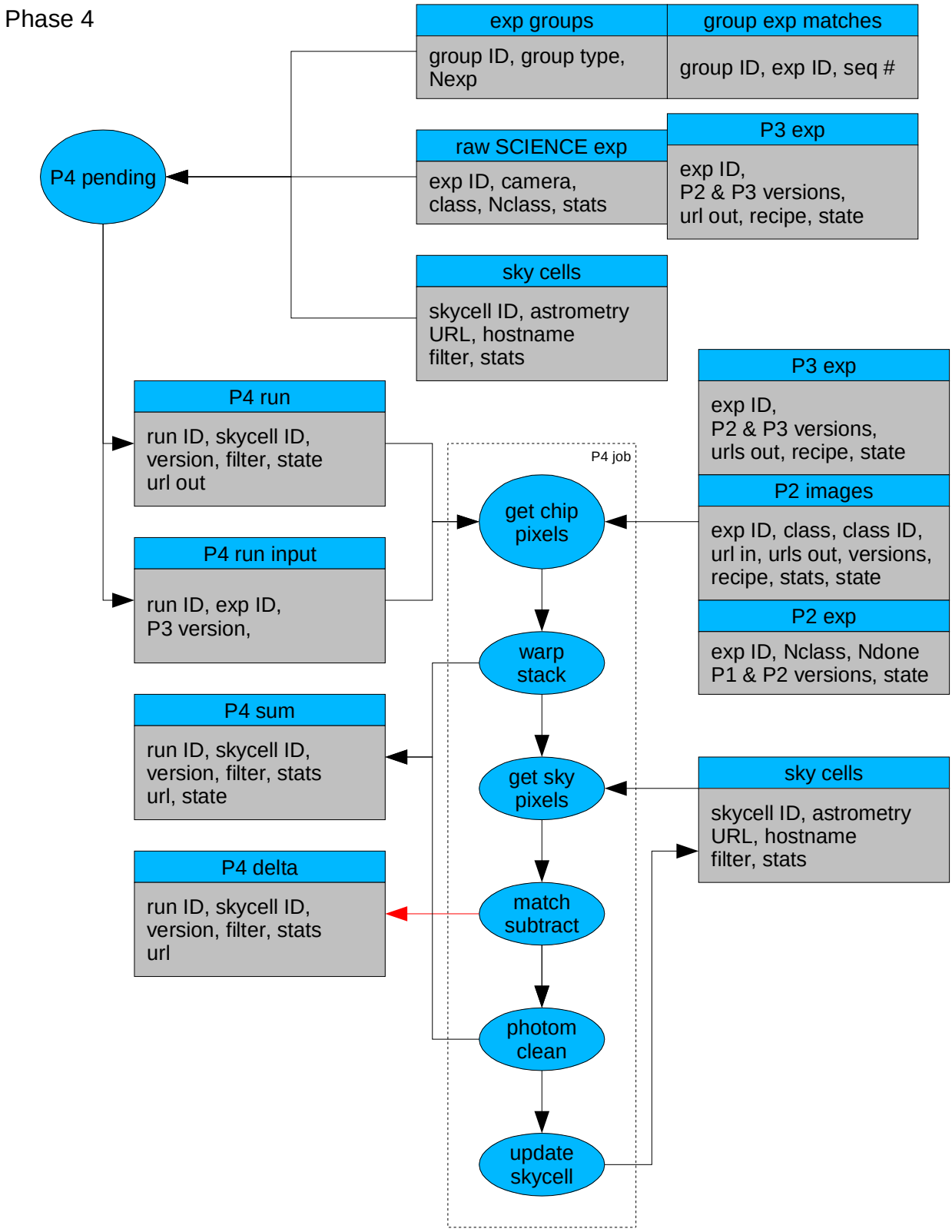

Figure 7: Phase 4 Tasks

results. The resulting photometry may be used to construct the new, corrected flat-field images, and the processing of the same images using the new flat-field images may be sent to the master DVO database.

13 Basic Detrend Creation

Figure 8 illustrates the tables needed for the generic detrend construction process, using the flat-field construction as an example. This diagram is somewhat more complex than the preceeding versions. In this diagram, both single jobs and multiple jobs are represented by the process elements (the blue ellipses). In some cases, more that one task will be needed to perform the function illustrated by a single process task. The complexity of this diagram is enhanced by the need for multiple iterations and both single chip and full mosaic processing. At the moment, the distinction between mosaic and single chip cameras is not specifically discussed. Finally, the triggers which initiate a specific detrend analysis are glossed over.

The detrend analysis is initiated by choosing a type of detrend image to be constructed and by specifying the criteria which will be used to select the input raw detrend frames for the construction. For example, these criteria could specify that all twilight flat images over a certain period of days, perhaps with restrictions on the flux levels or the time-from-sunset of the images. The detrend analysis run is given an ID (det ID) which will also be used to identify the resulting master detrend frame.

Given the definition of a master detrend run, the input exposures are selected from the raw detrend exposure table, and written to the input detrend exposure table. In the next step, the corresponding image files are selected from the table of raw image files. Since there will be a different set of input raw images for each attempt at creating a master detrend image, and since any given attempt may use some of the same input images as any other attempt, a separate table of input raw images is constructed.

Each of the input raw images may be pre-processed before it may be used to construct the detrend frame. For example, the input flat-field images should (probably) be dark- and bias-corrected before they are stacked. The information about these input processed images is written to the input images table. If no processing is needed, this step simply copies the appropriate information to the table, and points back to the raw image, rather than a processed version.

The input processed images are combined (stacked) to create a master detrend image for the particular data element defined by the image class (chip/cell/fpa). At this stage, not all input images should necessarily be included in the stack. If residual statistics have been measured for the input images (say, using a prior stack), then some of the input image may be excluded. The table of residual images is used to guide this process. The information describing the resulting master image is written to the master images table.

The statistics of the master detrend images must examined so that any necessary renormalizations may be performed. For example, after stacking the individual flat images, the resulting stacks must be renomalized to account for the different ranges of input image fluxes. This analysis is least-squares solution in which an appropriate scale is determined for each input exposure and a separate gain is determined for each of the chips or cells in the camera. This analysis can only performed after all image stacks (ie, for all chips) have been constructed. The resulting information is written to the table of master detrend frames.

Once the master detrend is constructed, the master detrend images may be used to construct residual images for each of the input images. These residual statistics, as well as the locations of the residual images and other related data products (jpeg thumbnails?) are written to the residual image table. Note the red arrow which by-passes the stack construction and merge steps and skips directly to the residual analysis. In some cases, it may be useful to have the input images confronted with an existing detrend image, and the resulting residual values used to guide the rest of the process. For example, in the flat-field analysis, applying an earlier flat can result in a very good first-pass rejection of poor input images. The logic to

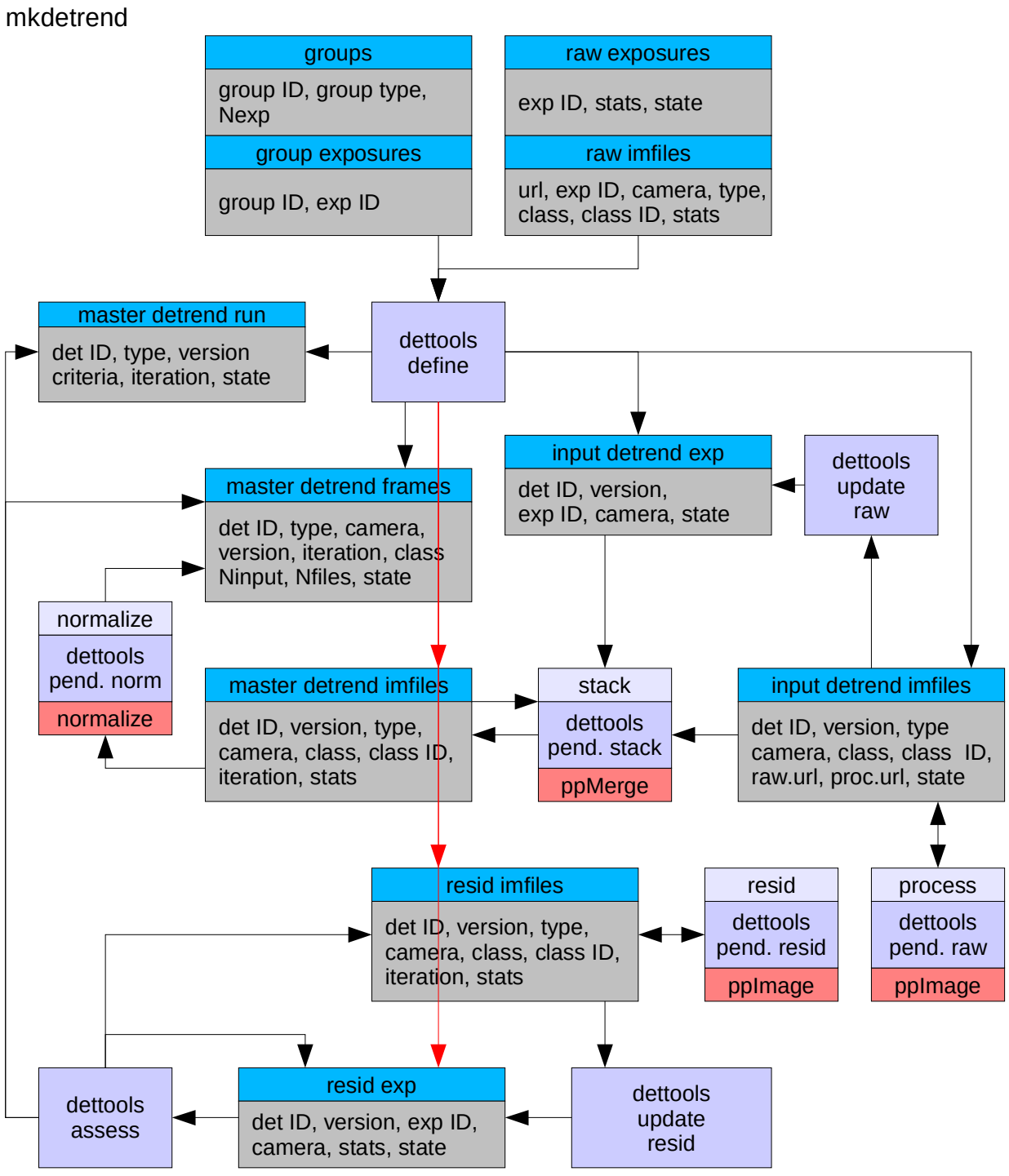

Figure 8: Detrend Creation Tasks

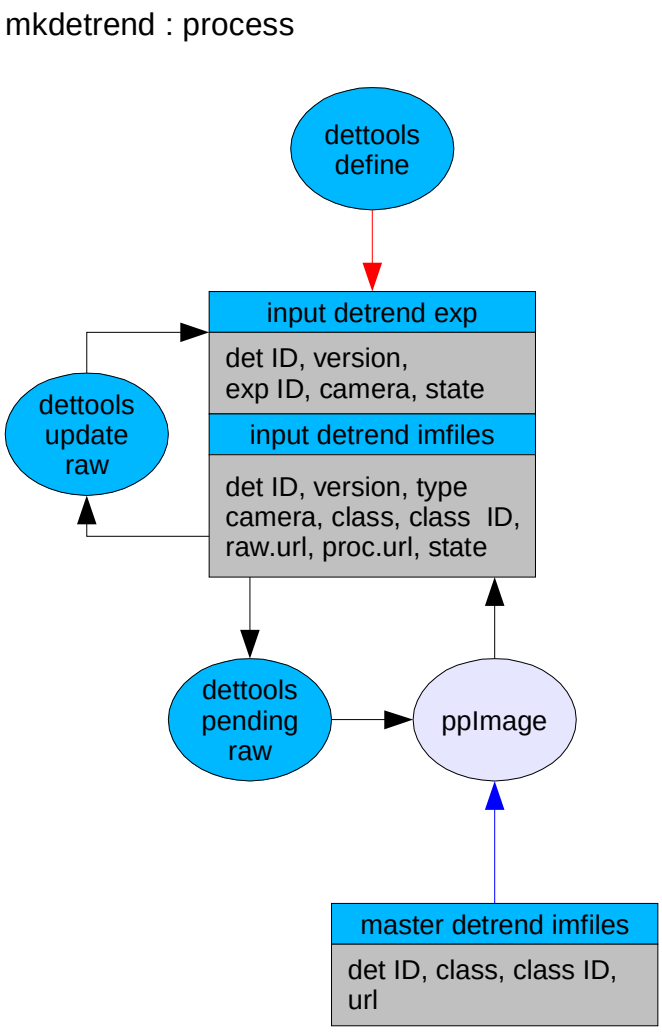

Figure 9: Detrend Creation : Process Tasks

make this leap must be part of PanTasks, since each of the individual blocks represent complete processing jobs.

Finally, the residual statistics from the complete mosaic (all input images, all chips) are used to assess the quality of the newly constructed master detrend image, and to potentially modify the selection of input images. This latter process is performed by marking the state of the residual images from this iteration. The stacking process always examines the state information for the residual images from the previous iteration, if it exists, when constructing the master stack. Once a master detrend frame has been judged of high enough quality, the state of the entry for the frame in the master detrend frames table is set to an appropriate value to tell the other routines that this image should be used as a master detrend. The exact choice of which master detrend frame is used for a given science image depends on the recipe along with information such as the time period used or the conditions used.

Note that, although this discussion focuses on the construction of flat-field images, the same structure should be capable of constructing the biases, dark, fringes, etc. In some cases, as noted above, the ‘process’ stage is a null operation.

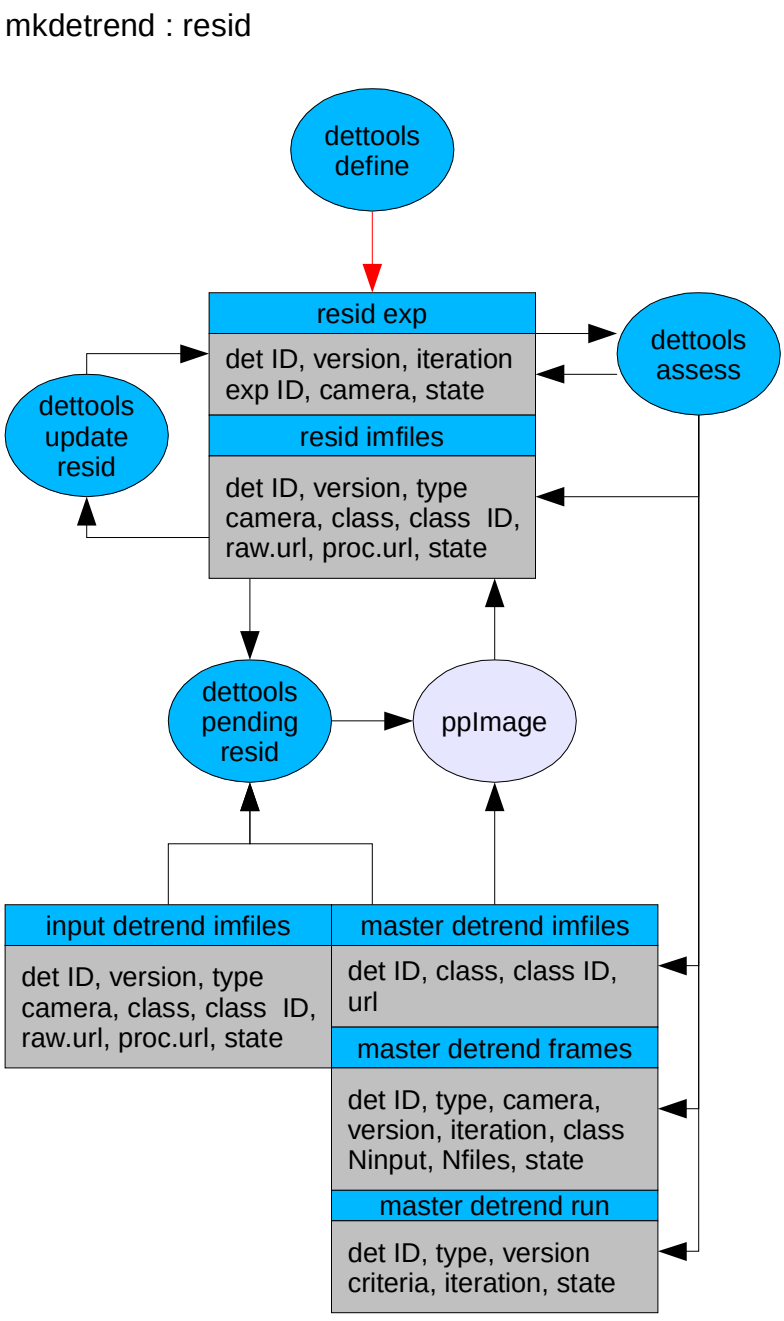

Figure 10: Detrend Creation : Residual Tasks

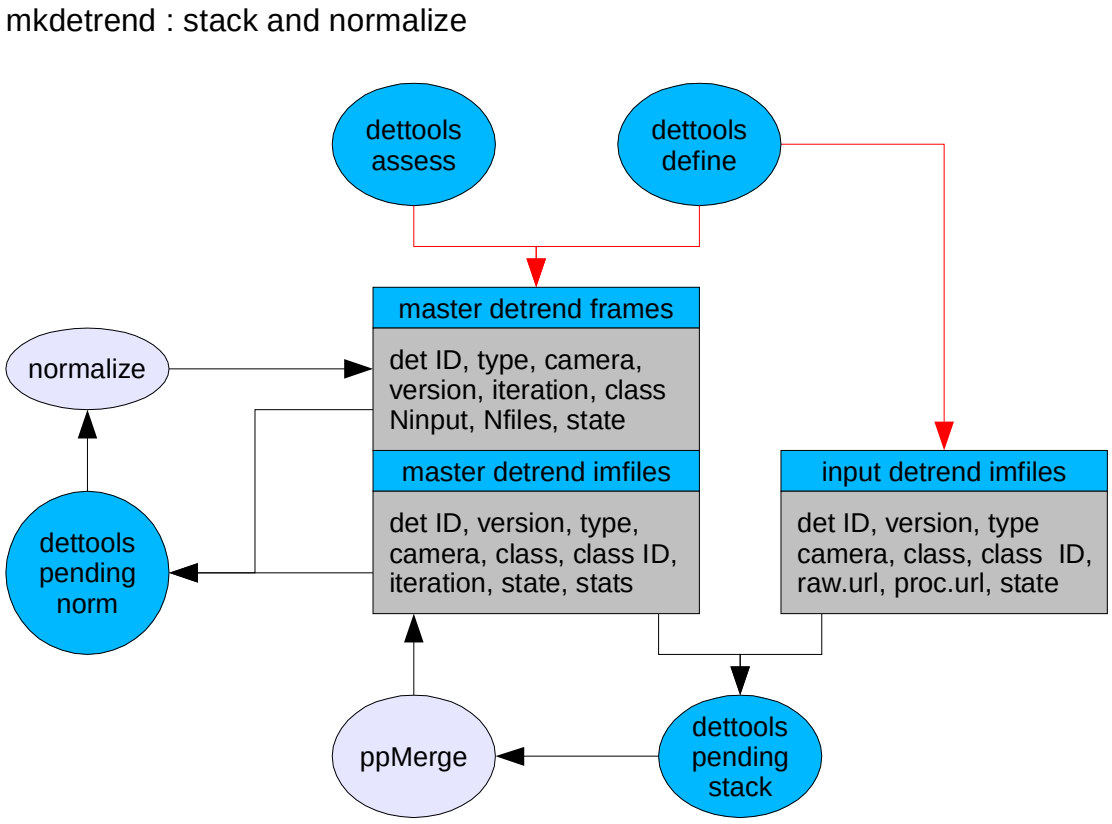

Figure 11: Detrend Creation : Stack and Norm

A IPP top-level commands

In this section, we define all the necessary top-level commands needed by the IPP to implement the data flow discussed in the previous sections. These commands are user commands, and are visible in the UNIX command space. The commands are discussed in the context of both the automatic processing and processing of individual entries. It is an important goal that the user commands should provide a simple-enough interface that they can be used without needing to have the entire infrastructure of the IPP in place to function. In practice, this means that data items which can be acquired from the Metadata DB tables can also be provided on the command line and/or in the recipe file. In some cases, user commands are provided to allow a manual intervention beyond the automatic processing loops. This is particularly true of the `submit.Px` type of commands.

this section should be reconciled with the section in the SSDD which takes precedence (TBD)

```
copy.image
  input: url, exp ID, camera, class, class ID

  This program copies the image file from the given URL, updates the
  new image file table with the descriptive metadata (exp ID, camera,
  class, class ID), and notifies the external subsystem that the copy
  has succeeded.  The destination host for the image file is
  determined by selecting the class ID from the host-for-class table.

copy.table
  input: url, table ID, table type

  This program copies the metadata table file from the given URL,
  determines which IPP metadata table it corresponds to, adds the
  table rows to that metadata table, and notifies the external
  subsystem that the table copy has succeeded.

classify.image
  input: (url) or (exp ID + class ID)

  This program examines the header of the specified image file,
  determines the image type (detrend, science, etc), extracts a
  specific set of information from the header, and adds a new entry to
  the raw image files table.  It also checks the new exposure table
  for the corresponding exp ID, counts the files with the same exp ID
  that are in the raw image files table, and if they match, adds an
  entry to the raw exposure table for the appropriate exposure type.
  Depending on the camera format (mosaic / single chip), the process
  also adds an entry to either the P1 exposure table or the P2 image table.

Phase 0 commands:

p0search -pending :
  * examine the new.imfiles,new.exposures tables and select exposures ready for analysis
  * output is: (expID) (camera)

p0search -update (expID) (camera):
  * select a the corresponding images from the new.imfiles/new.exposures table
  * extract the specified header information
  * search the summit metadata db tables
  * write an entry to the raw.imfiles and raw.exposure tables
  * set the state on the new.imfiles,new.exposure tables
  * based on the camera config information;
    * add an entry to the p1.pending table (mosaic)
    - or
    * add an entry to the p2.pending table (single)

p0search -stats (expID) (camera):
  * select a specified image in the new.imfiles/new.exposures table
```

```
  * extract the specified header information
  * search the summit metadata db tables
  * report the image stats
  [-update without output to MDDB]

p0search -mkraw (expID) (camera):
  * select a specified image in the new.imfiles/new.exposures table
  * extract the specified header information
  * search the summit metadata db tables
  * write an entry to the raw.imfiles and raw.exposure tables
  * set the state on the new.imfiles,new.exposure tables

p0search -cleanup:
  * remove completed entries from the new.imfiles,new.exposure tables

** note : the division of -pending and -update allows separate processes
   to be examining the image headers and measuring some stats.  these
   jobs can be run via pcontrol to reduce the load on the PanTasks
   machines

Phase 1 pipeline tools:

p1search -define [constraints]:
  * examine the raw.exposures tables and select exposures matching the given criteria
  * add entries which are allowed (mosaic) to the p1.pending table

p1search -pending :
  * examine the p1.pending table and select exposures waiting for p1
  * output: lines consisting of:
    (expID) (p1version) (camera)

p1search -done :
  * select completed entries in the p1.pending table
  * move to the p1.done table
  * add new entry to the p2.pending tables

Phase 2 pipeline tools:

p2search -quick
  * search for images which match in raw.exp,raw.imfiles
  * output in format which can be used by ppImage pantasks script

p2search -define [options]
  * input: searches mddb:raw_exposures,raw_images
  * output: updates mddb:P2_exposures_pending,P2_images_pending
  * alternative output: identical to p2pending

p2search -pending
  * input: searches mddb:P2_exposures_pending,P2_images_pending
  * output: Nlines consisting of:
    (URL) (expID) (class)
  * options: ?

p2search -update
  * examine the imfiles and identify any completed exposures

p2search -done
  * add completed exposures to the p2.done tables
  * remove corresponding entries from the p2.pending table
  * send new entry to the pending p3 table

ppImage file://path/filename file://path/outroot -recipe (recipe)
ppImage neb://nebname neb://outroot -recipe (recipe)

restriction options:
  -time (start) (stop)
  -camera (camera)
  -region (ra,dec) (ra,dec)
```

```
Phase 3 pipeline tools:

p3search -define :
  * examine the raw.exposures tables and select exposures matching the given criteria
  * add entries which are allowed (mosaic) to the p3.pending table

p3search -quick :
  * examine the raw.exposures tables and select exposures matching the given criteria
  * return list of entries for p3 processing
  * output: lines consisting of:
    (expID) (p3version) (camera)

p3search -pending :
  * examine the p3.pending exposures table and select exposures waiting for p3
  * return list of entries for p3 processing
  * output: lines consisting of:
    (expID) (p3version) (camera)

p3search -done :
  * select completed entries in the p3.pending table
  * move to the p3.done table


mkdetrend tools

dettools -define [options]
 * define a new detRun, specifying the constraints, and adding it to
 the run table.  if the detRun ID already exists, creates a new
 version.  the initial state is set to START.  the iteration is set to
 0. also creates a new master detrend frame entry
 (detID.version.iteration define this frame uniquely).
 * select the input matching a given detRun.  selects the input
 exposures and the input files.

options :
  -ID ID : a free-form string; if not specified, a unique string is constructed
  -type type : bias dark (mask?) flat fringe (other?)
  -camera camera
  -filter filter
  -time start stop
  -exptime min max
  -airmass min max
  -expgroup groupID

  (-type and -camera are mandatory)
  (-filter is mandatory for 'light' types)
  (-exptime is mandatory for dark)

dettools -pending raw [-state state] [-outmode mode]
 * select the unprocessed input infiles
 - output of this program is used by pantasks to schedule the ppImage
 run on each image

dettools -pending resid [-state state] [-outmode mode]
 * select the residual images to be processed
 - output of this program is used by pantasks to schedule the ppImage
 run on each image

dettools -pending stack [-state state] [-outmode mode]
 * select the files to be stacked for a given detRun, if exposures are ready
 - output of this program is used by ppMerge to build a master stack

dettools -pending norm
 * select the master detrend frames & imfiles to be normalized

dettools -update raw
```

```
 * select the raw input exposures, count processed imfiles, update if done

dettools -update resid
 * select the resid exposures, count processed resid imfiles, update if done
 - for a given resid exposure, compile the results from the stacks for
 each chip and renormalize the output files as needed.

dettools -assess
 * for a given master detrend frame, compile the results from the
 residual exposures and assess the validity of the input exposures and of
 the complete stack.  if too many input images are rejected, the
 affect the master detrend frame state.  if input images are rejected
 and a new set of master stacks should be made, create a new master
 detrend image, incrementing the iteration by one.

** NOTE the sequence is:

   process, stack, merge, residual, assess
              ^-------------------------<

   IF we have no relevant detrend image for comparison.  otherwise,
   the sequence should skip from process to residual on the first
   pass, with a lower rejection threshold for the first assess pass.
   the tools above allow either option; it is the choice of state
   after process that determines which happens next.



States:
 input detrend exposures:
  RAW
  PROCESSED

 input detrend imfiles:
  RAW
  PROCESSED

 master detrend frames
  RAW
  NORMALIZED
  MKRESID
  SUCCESS
  FAILURE
  RETRY
  (also has NEW/PRIOR flag)

 master detrend imfiles
  NEW (not yet created)
  RAW (created, but not normalized)
  NORMALIZED

 resid exposure
  RAW
  PROCESSED
  ASSESSED

 resid imfiles:
  RAW
  PROCESSED

 master detrend run:
  NEW
  DONE



older descriptions:
```

```
Phase.1
  input: exp ID

  This program determines the chips which correspond the exposure,
  loads the guide stars (if they exist) or reads the pixels around
  bright stars (if the guide stars do not exit.  It determines the
  centroids for these stars, reads in the astrometric reference stars
  in the field, matches observed stars to reference stars, determines
  an astrometric solution, using the default telescope / camera model
  as the starting point, and writes out the result to the P1 output
  file, along with the FITS table of observed star measurements (X, Y,
  inst mag).  It writes the summary statistics back to the P1 exposure
  table, an new entry in the P2 exposure table, and entries for each
  of the image files in the P2 images table.

Submit.P2
  input: exposure selection criteria

  This program uses the provided selection criteria (eg, exp ID, time
  range + filter, etc) and selects the corresponding exposures and
  image files.  If the output is specified to the database, it creates
  new entries in the P2 images and P2 exposure tables, incrementing
  the version fields if these exposure already exist.  If the output
  is specified to stdout / file, it writes the corresponding
  information in a human (and/or machine) readable format.  Note: it
  is not necessary that a P2 exposure defined by this tool include all
  possible or available image files.  Note also: this program must
  validate that the selected exposures have completed Phase 1, or are
  not required to complete Phase 1.

Phase.2
  input: (url) or (exp ID + class ID)

  This program reads in the selected image file, determines the
  matching detrend images, detrends the image data, performs object
  detection and analysis, loads the corresponding
  astrometric/photometric reference data, determines the astrometric
  and photometric calibrations.  It writes out the detrended image,
  the corresponding mask, and a FITS table of the measured objects,
  with the astrometric parameters in the FITS table header.  It
  updates the P2 images table with the statistics of the analysis, and
  updates the P2 exposure table status.  If this analysis is the last
  of the Nclass P2 image analyses to be done on this P2 exp ID and
  version, then the program makes a new entry in the P3 exposure
  table.

Submit.P3
  input: exposure selection criteria

  This program uses the provided selection criteria (eg, exp ID, time
  range + filter, etc) and selects the corresponding exposures which
  have succeeded in Phase 2 (a specific P2 version may be specified).
  If the output is specified to the database, it creates new entries
  in the P3 exposure table, incrementing the version field if this
  exposure already exists.  If the output is specified to stdout /
  file, it writes the corresponding information in a human (and/or
  machine) readable format.

Phase.3
  input: (exp ID) or (list of P2 object files)

  This program finds the collection of P2 object files for the
  specified exposure, reads in the object data and astrometric
  calibration terms, loads the corresponding astrometric/photometric
  reference data, determines the improved astrometric and photometric
  calibrations.  It writes out a single FITS table of the measured
  objects, with the astrometric parameters in the FITS table header,
  and astrometric calibration data in a new astrometry file for this
```

```
  exposure.  It updates the P3 exposure table with the statistics of
  the analysis

Submit.P4
  input: selection criteria

  This program uses the provided selection criteria (eg, exp ID list,
  observation group ID, sky region + time range + filter, etc) and
  selects the corresponding exposures which have succeeded in Phase 3
  (or Phase 2 if Phase 3 is not needed for this camera / recipe).  It
  determines the matching sky cells which are overlapped by the
  selected exposures.  If the output is specified to the database, it
  creates new entries in the P4 run table and the P4 run input table.
  If the output is specified to stdout / file, it writes the
  corresponding information in a human (and/or machine) readable
  format.

Phase.4
  input: (run ID) or (sky cell + list of P2 / P3 image files)

  This program uses the skycell and the list of input exposures to
  find the collection of image files which overlap the given skycell.
  It then reads the image pixels, warps them to match the skycell
  geometry, and stacks the image pixels.  It reads the pixels from the
  skycell and performs the image difference analysis, it photometers
  the difference image, cleans the input summed image on the basis of
  the detections, and photometers the input summed image.  It writes
  out the statistics of the analysis to the P4 run table, the P4 sum
  table, and the P4 delta table.  If requested, it improves the
  signal-to-noise in the skycell image, and updates the skycell table.

Detrend.init
  input: (detrend creation criteria)

  This program adds a new entry to the master detrend run table based
  on the given criteria.  The criteria may define a time range for the
  input images, a detrend type, flux ranges, a filter, an observation
  group ID.  The new master detrend run is created with the iteration
  set to 00 and the state set to 'new'.

Detrend.get.input.list
  input: det ID, version

  This program uses the selection criteria defined by the master
  detrend run (eg, exp ID list, observation group ID, time range, type
  + filter, etc) to select the corresponding detrend exposures.  It
  creates a new entry in the master detrend frames table, with state
  'new' and iteration of 00.  It also writes the exposures to the
  input detrend exposure table.

Detrend.get.images
  input: exp ID

  This program identifies the detrend images which correspond to the
  selected detrend exposure ID and adds an entries for the image files
  in the input detrend images table, with state set to 'raw'.

Detrend.process
  input: url / detrend type

  This program performs the pre-processing needed by the selected
  detrend image in order to prepare it for combination with other
  detrend images.  When completed, the entry for this file in the
  input detrend images table is update to 'proc'.  The processing may
  be as simple as a null operation (eg, for a bias) or as complex as
  bias, dark, flat-field, renormalize (eg, for a fringe).

Detrend.stack
```

```
  input: det ID, version, iteration, class ID?

  This program performs the image stacking for a single image class
  ID.  The stacking process may depend on the detrend type (different
  for bias from flat from fringe).  The resulting statitics are
  written to the master detrend image table.  If this is the last of
  Nclass entries for the given master detrend frame, then the master
  detrend frames state is updated.

Detrend.merge
  input: det ID, version, iteration

  This program examines the results of the Nclass master detrend
  images and performs any necessary re-normalizations.  It also
  examines the statistics of the individual stacks and summarizes them
  in the master detrend frame table..

Detrend.residuals
  input:  det ID, version, iteration, class ID, url

  This program performs the detrending on the given processed input
  image using the corresponding detrend frame.  The residual image is
  saved, and an entry is written to the residual images table giving
  the image location and the residual statistics for this image.

Detrend.assess
  input: det ID, version, iteration

  This program examines the collection of residual image statistics
  and creates ensemble statistics for each residual exposure (NOTE: do
  we need a residual exposure table??).  It examines the ensemble of
  exposure statistics and determines if 1) the complete stack meets
  the success criteria and 2) which residual exposure do / do not meet
  the success criteria.  It updates the master detrend run table, the
  master detrend frame table, and the residual image / exposure table
  to note images which should be included / excluded in a future
  iteration.
```

B Metadata Database Tables used for IPP Job Flow

The tables presented below define in greater detail the contents of the Metadata tables shown in the figures above. In some cases, the quantities (eg, the analysis result statistics) are illustrative, not definitive. In certain tables, data is provided which is redundant (non-normal) for ease of use. In some cases, we may decide not to keep this redundant information. In cases where the choice to replicate the redundant information is uncertain, the field names are written in *italics*.

Table 2: Pending Image Files

Field Name	Datatype	Description	Examples
URL	string	file location	http://data/file001.fits
exp ID	string	exposure ID	654321o
camera	string	camera name	MegaPrime / GPC
class	string	file data class	Cell / Chip / FPA
class ID	string	identify for class	chip00 / cell0102
state	string	state of transfer?	ready / copied

Table 3: New Image Files

Field Name	Datatype	Description	Examples
URL	string	file location	neb://file001.fits
exp ID	string	exposure ID	654321o
camera	string	camera name	MegaPrime / GPC
class	string	file data class	Cell / Chip / FPA
class ID	string	identify for class	chip00 / cell0102

Table 4: Host for Class

Field Name	Datatype	Description	Examples
camera	string	camera name	MegaPrime / GPC
class ID	string	identify for class	chip00 / cell0102
hostname	string	preferred host computer	po01 / alala.ifa.hawaii.edu

Table 5: Raw Image Files

Field Name	Datatype	Description	Examples
URL	string	file location	neb://file001.fits
exp ID	string	exposure ID	654321o
camera	string	camera name	MegaPrime / GPC
class	string	file data class	Cell / Chip / FPA
class ID	string	identify for class	chip00 / cell0102
type	string	exposure type	bias / flat / science
Nreadout	int	number of readouts	1 / 5 / 100
Treadout	float	readout exposure time	1.0 (sec)
ccdtemp	float	detector temperature	90 (K)
background	float	data area median	5.0 (sec)
FHWM	float	average image quality	2.5 (arcsec)

note: stats below the line are measured, perhaps they go elsewhere?

Table 6: Pending Metadata Tables

Field Name	Datatype	Description	Examples
URL	string	file location	neb://file001.fits
table ID	string	unique table identifier	table-654321
table type	string	table content type	temps / skyprobe data
state	string	copied?	ready / copied

Table 7: Copied Metadata Tables

Field Name	Datatype	Description	Examples
URL	string	file location	neb://file001.fits
table ID	string	unique table identifier	table-654321
table type	string	table content type	temps / skyprobe data
state	string	copied?	ready / copied
time	int	table arrival time	13243413 (s)

Table 8: New Exposures

Field Name	Datatype	Description	Examples
exp ID	string	exposure ID	654321o
camera	string	camera name	MegaPrime / GPC
telescope	string	telescope name	CFHT / PS-1 / SP-1
class	string	file data class	Cell / Chip / FPA
Nfiles	int	number image files	64

Table 9: Raw SCIENCE Exposure

Field Name	Datatype	Description	Examples
exp ID	string	exposure ID	654321o
camera	string	camera name	MegaPrime / GPC
telescope	string	telescope name	CFHT / PS-1 / SP-1
class	string	file data class	Cell / Chip / FPA
Nfiles	int	number of image files	1 / 64
exptime	float	exposure time	10.0 (seconds)
filter.ID	string	filter ID (glass)	g.PS1.01
filter.Band	string	filter bandpass name	g
airmass	float	sec(zenith angle)	1.35
RA	float	boresite RA	125.01 (degrees)
DEC	float	boresite DEC	35.01 (degrees)
PA	float	FPA position angle	10.1 (degrees)
obstime.sec	float	observation start	123423422 (TAI)
obstime.nsec	float	observation start (nsec)	1234
telfocus	float	focus distance	1.50 (mm)
FPAtemp	float	FPA temperature	90 (K)
dometemp	float	dome air temperature	290 (K)
airtemp	float	outside air temperature	285 (K)
mirrortemp	float	primary mirror temp	286 (K)
teltemp	float	telescope structure temp	283 (K)
group ID	string	observation group name	target-seq-01
group seq	int	sequence number in group	3
background	float	average of file skies	5.0
dbackground	float	stdev of file skies	1.0
FHWM	float		

Table 10: Raw Detrend Exposures

Field Name	Datatype	Description	Examples
exp ID	string	exposure ID	654321o
camera	string	camera name	MegaPrime / GPC
telescope	string	telescope name	CFHT / PS-1 / SP-1
type	string	exposure type	bias / domeflat / dark
class	string	file data class	Cell / Chip / FPA
Nfiles	int	number of image files	1 / 64
exptime	float	exposure time	10.0 (seconds)
filter.ID	string	filter ID (glass)	g.PS1.01
filter.Band	string	filter bandpass name	g
Alt	float	boresite Alt	125.01 (degrees)
Az	float	boresite Az	35.01 (degrees)
PA	float	FPA position angle	10.1 (degrees)
obstime.sec	float	observation start	123423422 (TAI)
obstime.nsec	float	observation start (nsec)	1234
telfocus	float	focus distance	1.50 (mm)
FPAtemp	float	FPA temperature	90 (K)
dometemp	float	dome air temperature	290 (K)
airtemp	float	outside air temperature	285 (K)
mirrortemp	float	primary mirror temp	286 (K)
teltemp	float	telescope structure temp	283 (K)
group ID	string	observation group name	target-seq-01
group seq	int	sequence number in group	3
callamp.state	string	calibration lamp state	on / off
callamp.mode	string	calibration lamp mode	continuum / line / ??
background	float	average of file back	5.0
dbackground	float	stdev of file back	1.0
FHWM	float		
hline			

Table 11: P1 Exposures

Field Name	Datatype	Description	Examples
exp ID	string	exposure ID	654321o
camera	string	camera name	MegaPrime / GPC
version	int	P1 version number	01
recipe	string	analysis recipe name	basic / flattest
state	string	P1 analysis state	new / done
output	string	location of output file	rootname.P1.01.smf
P1-log	string	location of P1 logfile	rootname.P1.01.log
Nstars	int	number of astrom stars	50
sigma.X	float	astrom scatter in X	1.0 (arcsec)
sigma.Y	float	astrom scatter in Y	1.2 (arcsec)
Mcal	float	nominal zeropoint	25.2 (mag)
Moff	float	measure ZP offset	0.5 (mag)
dMoff	float	scatter in Moff	0.1 (mag)

Table 12: P2 Exposures

Field Name	Datatype	Description	Examples
exp ID	string	exposure ID	654321o
class	string	file data class	Cell / Chip / FPA
Nfiles	int	number of P2 image files	3
Ndone	int	number completed	2
P1 version	int	P1 version number	01
P2 version	int	P2 version number	01
state	string	P2 analysis state	new / done / fail?

The P2 analysis produces a reduced (P2) output image, a table of (P2) measured objects, and astrometric calibration terms. The output objects are stored as a FITS table. The astrometric solution for the image is stored in the header of the output object file. Interesting statistics from the analysis are written to the MDDB table. Additional information describing the analysis is written to the log file in a machine-readable format. Additional entries can be added to the MDDB table if we find they are useful for understanding the processing results.

Table 13: P2 Images

Field Name	Datatype	Description	Examples
exp ID	string	exposure ID	654321o
class	string	file data class	Cell / Chip / FPA
class ID	string	identify for class	chip00 / cell0102
recipe	string	analysis recipe name	basic / flattest
state	string	P2 analysis state	new / done / fail?
input.url	string	file location	rootname.fits
output.image.url	string	file location	rootname.P2.01.fits
output.obj.url	string	file location	rootname.P2.01.smf
output.log.url	string	file location	rootname.P2.01.log
bias	float	measured bias value	5.0
dbias	float	bias residual scatter	5.0
fringe	float	fringe amplitude	10.0
dfringe	float	fringe scatter	10.0
sky	float	reduced background	5.0
dsky	float	background scatter	5.0
Nstars	int	number of astrom stars	50
sigma.X	float	astrom scatter in X	1.0 (arcsec)
sigma.Y	float	astrom scatter in Y	1.2 (arcsec)
Mcal	float	nominal zeropoint	25.2 (mag)
Moff	float	measure ZP offset	0.5 (mag)
dMoff	float	scatter in Moff	0.1 (mag)
runtime	float	processing time	20.2 (sec)

note: the stats below the line are examples to be extended

Table 14: P3 Exposures

Field Name	Datatype	Description	Examples
exp ID	string	exposure ID	654321o
class	string	file data class	Cell / Chip / FPA
Nfiles	int	number of P2 image files	3
Ndone	int	number completed	2
P1 version	int	P1 version number	01
P2 version	int	P2 version number	01
P3 version	int	P3 version number	01
state	string	P3 analysis state	new / done / fail?
recipe	string	analysis recipe name	basic / flattest
output	string	location of output file	rootname.P3.01.smf
P3-log	string	location of P3 logfile	rootname.P3.01.log
Nstars	int	number of astrom stars	50
sigma.X	float	astrom scatter in X	1.0 (arcsec)
sigma.Y	float	astrom scatter in Y	1.2 (arcsec)
Mcal	float	nominal zeropoint	25.2 (mag)
Moff	float	measure ZP offset	0.5 (mag)
dMoff	float	scatter in Moff	0.1 (mag)
runtime	float	processing time	20.2 (sec)

Table 15: Master Dark Frames

Field Name	Datatype	Description	Examples
det ID	string	detrend frame ID	654321o
version	int	version of det frame	02
label	string	descriptive name	basic
recipe	string	creation recipe	basic
type	string	detrend type	bias / dark / flat
class	string	file data class	Cell / Chip / FPA
Nfiles	int	number of files	2
exptime	float	exposure time	5.0
Ninput	int	number of input frames	5
sigma	float	stack residual stats	2.0

Table 16: Master Flat Frames

Field Name	Datatype	Description	Examples
det ID	string	detrend frame ID	654321o
version	int	version of det frame	02
label	string	descriptive name	basic
type	string	detrend type	bias / dark / flat
class	string	file data class	Cell / Chip / FPA
Nfiles	int	number of files	2
filter.ID	string	filter ID (glass)	g.PS1.01 / none
filter.Band	string	filter bandpass name	g / none
Ninput	int	number of input frames	5
sigma	float	stack residual stats	2.0
Ntotal	float	ave.total input counts	1e6

Table 17: Master Fringe Frames

Field Name	Datatype	Description	Examples
det ID	string	detrend frame ID	654321o
version	int	version of det frame	02
label	string	descriptive name	basic
type	string	detrend type	bias / dark / flat
class	string	file data class	Cell / Chip / FPA
Nfiles	int	number of files	2
filter.ID	string	filter ID (glass)	g.PS1.01 / none
filter.Band	string	filter bandpass name	g / none
Ninput	int	number of input frames	5
sigma	float	stack residual stats	2.0
Ntotal	float	ave.total input counts	1e6
fringe	float	fringe amplitude	52.0
dfringe	float	fringe stdev	5.0
airmass.min	float	minimum airmass	1.00
airmass.max	float	maximum airmass	1.20

Table 18: Master Detrend Image Files

Field Name	Datatype	Description	Examples
URL	string	file location	neb://file001.fits
det ID	string	detrend frame ID	654321o
camera	string	camera name	MegaPrime / GPC
class	string	file data class	Cell / Chip / FPA
class ID	string	identify for class	chip00 / cell0102
type	string	exposure type	bias / flat / science

Table 19: Master Detrend Run

Field Name	Datatype	Description	Examples
det ID	string	detrend frame ID	654321o
camera	string	camera name	MegaPrime / GPC
version	int	version of det run	02
type	string	detrend type	bias / dark / flat
criteria	string	image selection criteria	filter flags
state	string	analysis state	new / done

Table 20: Input Detrend Exp

Field Name	Datatype	Description	Examples
det ID	string	detrend frame ID	654321o
camera	string	camera name	MegaPrime / GPC
version	int	version of det run	02
exp ID	string	exposure ID	654321o
state	string	analysis state	new / raw / proc

Table 21: Input Detrend Image Files

Field Name	Datatype	Description	Examples
raw.URL	string	location of raw file	neb://file001.fits
proc.URL	string	location of proc file	neb://file001.fits
det ID	string	detrend frame ID	654321o
version	int	version of det run	02
camera	string	camera name	MegaPrime / GPC
type	string	exposure type	bias / flat / science
class	string	file data class	Cell / Chip / FPA
class ID	string	identify for class	chip00 / cell0102
state	string	analysis state	new / raw / proc

C Summit Copy Tasks

```
# identify the images ready for copy
# new entries are added to queue newImages
# need to compare the new list with the ones already being processed
task        new.images
  command      new.images
  host         local

  periods      -poll 1
  periods      -exec 1
  periods      -timeout 30

  # success
  task.exit    0
    local i j Nstdout Nimages
```

```
    # compare output with new.image queue
    # keep only new entries
    queuesize stdout -var Nstdout
    for i 0 $Nstdout
      queuepop stdout -var line
      queuepush newImages -uniq -key 0 "$line"
    end
  end

  # locked list
  task.exit    1
    echo       "new.images: exec failure"
    $new.image.failure ++
  end

  # default exit status
  task.exit    default
    echo       "new.images: unknown exit status: $EXIT"
    $new.image.failure ++
  end

  # operation times out?
  task.exit    timeout
    echo       "new.images: timeout"
    $new.image.failure ++
  end
end

# copy new images, sending job to desired host
task        copy.images
  periods      -poll 0.5
  periods      -exec 0.1
  periods      -timeout 5

  task.exec
    queuesize  newImages -var N
    if ($N == 0) break
    if ($network == 0) break

    queuepop newImages -var line
    list tmp -split $line
    $filename   = $tmp:0
    $chip       = $tmp:1
    $state      = $tmp:2
    if ($state == new)
      # copy this image
      queuepush newImages -replace -key 0 "$filename $chip run"
    else
      # ignore this image
      queuepush newImages -replace -key 0 "$filename $chip $state"
      break
    end
    $host = `chip.host $chip`
    host $host
    command copy.image $filename $chip
  end

  # success
  task.exit    0
    # echo "done copy..."
    queuepop stdout -var line
    list tmp -split $line
    $filename   = $tmp:0
    $chip       = $tmp:1
    exec mark.image $filename
    queuepush newImages -replace -key 0 "$filename $chip copy"
  end
```

```
  # default exit status
  task.exit    default
    echo       "new.images: unknown exit status: $EXIT"
    $new.image.failure ++
  end

  # operation timed out?
  task.exit    timeout
    echo       "new.images: timeout"
    $new.image.failure ++
  end
end
```

Table 22: Input Detrend Resid Image Files

Field Name	**Datatype**	**Description**	**Examples**
image.URL	string	location of resid image	neb://file001.fits
thumbnail.URL	string	location of resid jpeg	neb://file001.fits
det ID	string	detrend frame ID	654321o
version	int	version of det run	02
iteration	int	det creation iteration	03
camera	string	camera name	MegaPrime / GPC
type	string	exposure type	bias / flat / science
class	string	file data class	Cell / Chip / FPA
class ID	string	identify for class	chip00 / cell0102
state	string	analysis state	new / raw / proc
scatter	float	residual scatter	1.0