UNIVERSITY OF HAWAII AT MĀNOA
Institute for Astrononmy

**Pan-STARRS Project Management System**

# Pan-STARRS Image Processing Pipeline
# Configuration Guide

## IPP CG

| | |
|---|---|
| **Grant Award No.** | **: F29601-02-1-0268** |
| **Prepared For** | **: Pan-STARRS PMO** |
| **Prepared By** | **: Paul Price** |

| | |
|---|---|
| **Document No.** | **: PSDC-430-???** |
| **Document Date** | **: June 19, 2007** |
| **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

# Revision History

| Revision Number | Release Date | Description |
|---|---|---|
| DR | 2006 Oct 17 | Draft |

## Referenced Documents

**Internal Documents**

| Reference | Title |
|---|---|
| PSDC-430-005 | Pan-STARRS PS-1 IPP Software Requirements Specification |
| PSDC-430-011 | Pan-STARRS PS-1 IPP System/Subsystem Design Description |
| PSDC-430-012 | Pan-STARRS PS-1 IPP Modules SDRS |

**External Documents**

| Reference | Title |
|---|---|

# Contents

# List of Figures

# 1 Introduction

This document describes the configuration files used for the Pan-STARRS Image Processing Pipeline (IPP). Configuration files are used to provide parameters to the IPP programs that can be ingested at run-time, and easily changed ("configured") as circumstances require.

PSLib defines a `psMetadata` structure which can carry labeled data of arbitrary types. The associated functions implemented in PSLib consist of tools to manipulate and extract data from `psMetadata` collections. A particular application of the `psMetadata` structure within PSLib is to carry the data from a FITS header. Other general-purpose information is also carried with the structure. Functions are available to read/write a `psMetadata` collection from/to a text-based configuration file using a human-readable syntax. We therefore use these "metadata configuration" (MDC) files as the basis for our configuration files. When referring to entries in an MDC file, we use the convention in this document that `NAME(TYPE)` refers to the item called `NAME`, with type `TYPE`.

The IPP uses configuration parameters on four levels:

- Options for the particular site installation of the pipeline: the *site*;
- Options specifying the instrument setup: the *camera*;
- Options specifying the format of the FITS file: the *format*; and
- Options specifying the particular parameter choices that affect the details of an analysis: the *recipe*.

Note that these are arranged in an hierarchical order, with the site configuration being the most general, and the recipe configuration the most specific. For example, not all sites will have to deal with all cameras, and different cameras may require different recipes at different times according to their particular quirks, analysis experimentations, or their evolution. Once the camera configuration is known under a particular circumstance, the appropriate recipe may be selected.

# 2 Site configuration

## 2.1 Location

The location for the site configuration file itself is configurable. The filename can be specified by the following:

2.1.1 The `-site` option[1] on the command line if provided;

2.1.2 The environment variable `PS_SITE`, if defined; or

2.1.3 `$HOME/.ipprc` otherwise.

## 2.2 Contents

The site configuration carries information particular to the site, that is, to the global setup of the IPP. Its responsibility is to configure the directories, database, cameras, recipes, and psLib.

[1] `-site` is used for C programs. For Perl programs, we use the `Getopt::Long` module which requires us to use `--site`

### 2.2.1 Directories

The PATH(STR) entry gives a colon-delimited list of paths that are to be searched for configuration files. This allows the user to neglect a long leading path when specifying filenames within the site and camera configuration files.

The DATAPATH(METADATA) entry contains a series of symbolic links (of tyoe STR) to data directories. This allows data to be moved to a different system (with different directory structure) without having to search and replace all paths within the database; and also to juggle multiple projects using the same configuration file. In the Perl components of the IPP, we use path://DIR to mean "look up DIR in the DATAPATH to get the directory".

The DATAPATH(METADATA) entry supplies a list of directories which may be used as aliases. Filenames which have the form path://PATH/remainder will be converted to a UNIX path by stripping the path:// component and replacing PATH with its value from the DATAPATH list. This allows the database to be moved to a different system (with different directory structure) without having to search and replace all paths.

### 2.2.2 Database setup

The following four entries give the required information for psDBInit() to establish a connection with the database.

DBSERVER(STR) specifies the database host name.

DBNAME(STR) specifies the database name.

DBUSER(STR) specifies the database user name.

DBPASSWORD(STR) specifies the database password. **DBPASSWORD is an insecure method of storing what might be sensitive information. This is to be revised in the future. (TBD)**

### 2.2.3 Cameras

The CAMERAS(METADATA) item contains a list of cameras, with their corresponding camera configuration files (of type STR). The order is significant, since this is the order in which cameras will be compared with FITS headers (see pmConfigCameraFromHeader) — the first matching camera found will be used. For this reason, it is useful to leave a simple catch-all camera configuration at the end, as a fall-back option.

### 2.2.4 Recipes

Recipe configuration files that are common for all cameras may be listed in the RECIPES(METADATA). The list consists of the recipe symbolic names with corresponding filename (of type STR).

### 2.2.5 psLib

TIME(STR) specifies the location of the time configuration file for psLib. This is not too important, since the original installation location is known by psLib.

LOGLEVEL(S32) specifies the logging level for psLogMsg.

LOGFORMAT(STR) specifies the log format (see psLogSetFormat).

LOGDEST(STR) specifies the log destination (see psMessageDestination for acceptable formats). We recommend setting this to STDERR to avoid problems associated with higher-level programs attempting to parse unintended input from stdout.

TRACEFORMAT(STR) specifies the trace format (see psTraceSetFormat).

TRACEDEST(STR) specifies the trace destination (see psMessageDestination for acceptable formats). We recommend setting this to STDERR to avoid problems associated with higher-level programs attempting to parse unintended input from stdout.

TRACE(METADATA) gives a list of trace facilities and their accompanying levels (of type S32); see psTraceSetLevel. We recommend including at least err, with level 10, since this will print all error messages. Other useful traces to set are psModules.camera for camera (and especially pmFPAfile) operations; and psLib.db for database lookups.

## 2.3 Example

```
### Example .ipprc file

PATH            STR     .:/my/home/.ipp # Default search path for configuration files

# Place your data directories here and refer to as path://PATH/remainder
DATAPATH METADATA
HERE STR /data/my/host/
THERE STR /data/other/host/
END

### Database configuration
DBSERVER        STR     localhost               # Database host name (for psDBInit)
DBNAME          STR     my_database             # Database name (for psDBInit)
DBUSER          STR     my_name                 # Database user name (for psDBInit)
DBPASSWORD      STR     my_password             # Database password (for psDBInit)

### Setups for each camera system
CAMERAS         METADATA
CTIO_MOSAIC2 STR ctio_mosaic2/camera.config # CTIO MOSAIC2 camera, for ESSENCE
        MEGACAM         STR     megacam/camera.config           # Megacam, on CFHT
        GPC1            STR     gpc1/camera.config              # Pan-STARRS GigaPixel Camera 1
        ISP             STR     isp/camera.config               # Pan-STARRS Imaging Sky Probe
        SIMPLE          STR     simple/camera.config            # Simple single-chip camera
END

### psLib setup
TIME            STR     pslib/psTime.config     # Time configuration file
LOGLEVEL        S32     9                       # Logging level; 3=INFO
LOGFORMAT       STR     THLNM                   # Log format
LOGDEST         STR     STDERR                  # Log destination
TRACEDEST       STR     STDERR                  # Trace destination
TRACEFORMAT     STR     THLNM                   # Trace format
TRACE           METADATA                        # Trace levels
        err                 S32     10
#       psLib.db            S32     10
#       psModules.camera    S32     10
END

RECIPES         METADATA                # Site-level recipes
PPIMAGE STR recipes/ppImage.config  # Image reduction
PPMERGE STR recipes/ppMerge.config # Image combination
  PPSTATS STR recipes/ppStats.config # Image statistics
PPSTATS_PHASE0 STR recipes/ppStats_phase0.config # Image statistics for Phase 0
PSPHOT STR      recipes/psphot.config # Photometry
PSASTRO STR recipes/psastro.config # Astrometry
```

```
END
```

# 3 Camera configuration

The Focal Plane hierarchy (`pmFPA, pmChip, pmCell, pmReadout`) is explained in more detail in the psModules SDRS (PSDC-430-012). The top level, an FPA, contains one or more chips, which correspond to a contiguous piece of silicon. A chip contains one or more cells, which correspond to a single amplifier. A cell contains one or more readouts, which correspond to individual reads of the detector.

The purpose of the camera configuration is to define the contents of the Focal Plane hierarchy, and to define parameters that are particular to the camera.

## 3.1 Location

The camera configuration may be specified by the `-camera` option on the command line. Failing that, locations for all known camera configuration files are specified within the site configuration, under the `CAMERAS(METADATA)`, which lists cameras by name, with their corresponding configuration file. Note that the `PATH(STR)` in the site configuration defines the search path for these files.

## 3.2 Contents

The camera configuration specifies information particular to the data from a particular camera. Note that the data from a camera may be stored in different formats (e.g., one amplifier per extension, vs all amplifiers spliced together in the PHU). The camera configuration contains the formats, the camera description, filter translation table, observation type translation table, recipes, rejection levels and file rules.

### 3.2.1 Formats

`FORMATS(METADATA)` contains a list of formats for the camera, with their corresponding camera format configuration files (of type `STR`). The order is significant, since this is the order in which formats will be compared with FITS headers (see `pmConfigCameraFormatFromHeader`) — the first matching format found will be used.

### 3.2.2 Camera description

`FPA(METADATA)` contains a list of chips that comprise the focal plane array. The corresponding values, of type `STR` are a whitespace-delimited list of cells that comprise the chip. These chip and cell names are symbolic names, that needn't match any particular detail. The chip names must be unique within the FPA, and the cell names must be unique within the chip.

### 3.2.3 Filter translation table

`FILTER.ID(METADATA)` contains a list of filter names (generally those found within the FITS header; e.g., `r.MP9601`), with an abstract name to describe the filter (e.g., `r`), of type `STR`. This allows multiple descriptions of the same filter that may exist in the FITS headers to be resolved as the same thing.

#### 3.2.4 Observation type translation table

`OBSTYPE.TABLE(METADATA)` contains a list of observation types (generally those found within the FITS header; e.g., `ZERO`), with an abstract name to describe the observation type (e.g., `BIAS`). This allows multiple descriptions of the same observation type that may exist in the FITS headers to be resolved as the same thing (e.g., `BIAS`, `ZERO` and `PEDESTAL` can all be set to `BIAS`).

#### 3.2.5 Recipes

Recipe configuration files that are common for the camera may be listed in the `RECIPES(METADATA)`. The list consists of the recipe symbolic names with corresponding filename (of type `STR`).

#### 3.2.6 Rejection levels

`REJECTION(METADATA)` contains a list of rejection levels, by detrend type, for use in the detrend creation steps (see `detrend_reject_imfile.pl` and `detrend_reject_exp.pl`). For each detrend type, the following sub-components of a `METADATA`, each of type `STR`, are expected to be defined:

- `FILTER`: provides additional specificity for the rejection limits, allowing multiple limits for a single detrend type to be defined, with the search for the appropriate value being made with an additional qualifier. This is useful, for example, for flats, where some filters require looser rejection levels than for others. The use of this keyword needn't be restricted to wavelength filters, however. Note that detrend types listed more than once must be declared as `MULTI`. If the `FILTER` is `*` (or absent), then it will only match if no filter is provided for the search.
- `EXPECTED`: the expected value for the mean in residual images.
- `IMFILE.MEAN`: rejection level for the mean at the imfile level, after removing the expected value and taking the absolute value.
- `IMFILE.STDEV`: rejection level for the standard deviation at the imfile level.
- `EXP.MEAN`: rejection level for the mean at the exposure level, after removing the expected value and taking the absolute value.
- `EXP.STDEV`: rejection level for the standard deviation at the exposure level.
- `EXP.MEANSTDEV`: rejection level for the standard deviation of the mean at the exposure level.
- `ENSEMBLE.MEAN`: rejection level for an exposure within an ensemble of exposures, comparing the exposure to the mean; in terms of standard deviations.
- `ENSEMBLE.STDEV`: rejection level for an exposure within an ensemble of exposures, comparing the variance to the mean variance; in terms of standard deviations.

- `ENSEMBLE.MEANSTDEV`: rejection level for an exposure within an ensemble of exposures, comparing the standard deviation of the mean to the mean standard deviation of the mean; in terms of standard deviations. **Confusing enough? (TBD)**
- `IMFILE.SN`: rejection level for the signal-to-noise at the imfile level.
- `EXP.SN`: rejection level for the signal-to-noise at the exposure level.

Apart from `FILTER`, values that are set to zero are ignored.

Because the above values must be duplicated multiple times, it may be useful to define a type:

```
TYPE    LIMITS  FILTER  EXPECTED IMFILE.MEAN IMFILE.STDEV EXP.MEAN EXP.STDEV EXP.MEANSTDEV ENSEMBLE.MEAN
ENSEMBLE.STDEV ENSEMBLE.MEANSTDEV IMFILE.SN EXP.SN
```

### 3.2.7 File rules

**EAM to check and supplement this description. (TBD)**

The file rules are one of the most important aspects of the camera configuration, and one of the easiest to get wrong. When setting up a new camera configuration and getting errors (or worse, segmentation faults), check the file rules first. Try turning up the `psModules.camera` trace level to see what's going on.

`FILERULES(METADATA)` lists the different types of files used in the image processing, which specify how and when a file is read in and written out. The files usually are of two or three components, separated by a period (not for any particular reason except that's what's been adopted); the first part specifies the program the file will be used in, the second and third parts identify its role. For example, `PPIMAGE.INPUT` specifies the input file for `ppImage`; `PPIMAGE.OUTPUT.MASK` specifies the output mask file from `ppImage`.

#### 3.2.7.1 Replacements

Throughout the file rules, a syntax for defining strings from variables is used: curly brackets `{}` around an abstract name are replaced by the program to obtain the proper value. Supported abstract names are:

- `{OUTPUT}` — replaced with the output file root;
- `{CHIP.NAME}` — replaced with the chip name;
- `{CHIP.N}` — replaced with the chip number (printed `%02d`);
- `{CELL.NAME}` — replaced with the cell name;
- `{CELL.N}` — replaced with the chip number (printed `%02d`);
- `{EXTNAME}` — replaced with the extension name;
- `{FILTER}` — replaced with the filter name (without applying the `FILTER.ID` translation table);
- `{FILTER.ID}` — replaced with the filter identifier (after applying the `FILTER.ID` translation table);
- `{CAMERA}` — replaced with the instrument name (from `FPA.INSTRUMENT`);

- {INSTRUMENT} — replaced with the instrument name (from FPA.INSTRUMENT);
- {DETECTOR} — replaced with the detector name (from FPA.DETECTOR); and
- {TELESCOPE} — replaced with the telescope name (from FPA.TELESCOPE).

(More could potentially be added. If one you greatly desire is missing, please ask!)

#### 3.2.7.2 Redirections

Entries with type STR are treated as symbolic links to another line. For example, specifying:

```
PPIMAGE.OUTPUT    STR    PPIMAGE.OUTPUT.SPLIT
```

means that the program will look up PPIMAGE.OUTPUT.SPLIT in the place of PPIMAGE.OUTPUT. This allows a quick replacement if a different output format is desired (e.g., PPIMAGE.OUTPUT.MEF instead of PPIMAGE.OUTPUT.SPLIT).

#### 3.2.7.3 File types

Both the input and output file rules use file types. The currently supported file types are:

- IMAGE — image data in FITS image format (treated as F32);
- MASK — mask data in FITS image format (treated as U8);
- WEIGHT — weight data in FITS image format (treated as F32)
- FRINGE — fringe image with fringe tables (one for each cell) in FITS image format (image treated as F32);
- JPEG — image data in JPEG format (output only);
- CMP — object data in CMP format;
- CMF — object data in CMF format;
- RAW — object data in RAW format;
- SX — object data in SX format; and
- OBJ — object data in OBJ format.

**EAM to fill in details on the object formats. (TBD)**

#### 3.2.7.4 Inputs

It is useful to make the following TYPE declaration, which can be used for all input files:

```
TYPE   INPUT   FILENAME.RULE   FILENAME.XTRA   EXTNAME.RULE   EXTNAME.XTRA   DATA.LEVEL   FILE.TYPE
```

The components are:

- FILENAME.RULE — this specifies the rule for constructing the filename. Options for doing so are:
  - A simple filename, perhaps using the replacement syntax defined above;
  - @DETDB to look up the appropriate file using the detrend database (see §5.2.1); or
  - @FILES to indicate that the input file(s) will be specified on the command-line of the program.
- FILENAME.XTRA — **PAP is not entirely sure what this is for; it may be unnecessary. (TBD)**
- EXTNAME.RULE — This defines the extension name. **Is this true? PAP thinks the camera format does that; this may be unnecessary, or it may have to be tied into the camera format. (TBD)**
- EXTNAME.XTRA — **PAP is not entirely sure what this is for; it may be unnecessary. (TBD)**
- DATA.LEVEL — the level of the hierarchy at which the data is to be opened for reading. This should correspond to the level of the extension in the FITS file, or higher. There are some checks against the camera format that this is sensical, but don't bet your life on it just yet. This is an important setting to check if you're having problems.
- FILE.TYPE — the type of file, from the above list (§3.2.7.3).

#### 3.2.7.5 Outputs

It is useful to make the following TYPE declaration, which can be used for all output files:

```
TYPE   OUTPUT   FILENAME.RULE   FILENAME.XTRA   EXTNAME.RULE   EXTNAME.XTRA   FILE.LEVEL   DATA.LEVEL   FILE.TYPE   F
```

The components are:

- FILENAME.RULE — this specifies the rule for constructing the filename. You most likely want to include {OUTPUT} somewhere here; Pan-STARRS convention is that it goes at the front.
- FILENAME.XTRA — **PAP is not entirely sure what this is for; it may be unnecessary. (TBD)**
- EXTNAME.RULE — This defines the extension name. **Is this true? PAP thinks the camera format does that; this may be unnecessary, or it may have to be ties into the camera format. (TBD)**
- EXTNAME.XTRA — **PAP is not entirely sure what this is for; it may be unnecessary. (TBD)**
- FILE.LEVEL — the level of the hierarchy at which a file should be opened and the PHU written. This should correspond to the level of the PHU. There are some checks against the camera format that this is sensical, but don't bet your life on it just yet. This is an important setting to check if you're having problems.

- `DATA.LEVEL` — the level of the hierarchy at which an extension should be written. This should correspond to the level of the extensions in the FITS file, or higher. There are some checks against the camera format that this is sensical, but don't bet your life on it just yet. This is an important setting to check if you're having problems.
- `FILE.TYPE` — the type of file, from the above list (§3.2.7.3).
- `FILE.SAVE` — whether this type of file should be saved (`TRUE`) or not (`FALSE`).
- `FILE.FORMAT` — if the file format is to be changed, this is the name of the file format (from the `FORMATS` metadata). Otherwise, it is `NONE`.

## 3.3 Example

```
# "mcshort" is a MegaCam camera with only the central six chips --- it's faster than the entire FPA.
# Camera configuration file for mcShort: describes the camera

# File formats that we know about
FORMATS         METADATA
        RAW     STR     mcshort/format_raw.config
        SPLICE  STR     mcshort/format_spliced.config
        SPLIT   STR     mcshort/format_split.config
END

# Description of camera --- all the chips and the cells that comprise them
FPA     METADATA
        ccd12   STR     LeftAmp RightAmp
        ccd13   STR     LeftAmp RightAmp
        ccd14   STR     LeftAmp RightAmp
        ccd21   STR     LeftAmp RightAmp
        ccd22   STR     LeftAmp RightAmp
        ccd23   STR     LeftAmp RightAmp
END


# Lookup table to go from FPA.FILTER to abstract name for the filter
FILTER.ID       METADATA
   u.MP9301     STR u
   g.MP9401     STR g
   r.MP9601     STR r
   i.MP9701     STR i
   z.MP9801     STR z
   Zp           STR z
   Zprime       STR z
   Ha.MP7605    STR Ha
   Halpha       STR Ha
   Haalpha.on   STR Ha
   HaOFF.MP7604 STR HaOff
END

# Lookup table to go from FPA.OBSTYPE values to abstract name for the exposure type
OBSTYPE.TABLE METADATA
  bias     STR BIAS
  zero     STR BIAS
  dark     STR DARK
  flat     STR SKYFLAT
  skyflat  STR SKYFLAT
  domeflat STR DOMEFLAT
  object   STR OBJECT
  science  STR OBJECT
END

# Recipe options
```

```
RECIPES METADATA
# Other recipes
        PSPHOT          STR     megacam/psphot.config           # psphot details
        PSASTRO         STR     megacam/psastro.config          # psastro details
PPSTATS STR megacam/ppStats.config # ppStats recipe
PPIMAGE         STR     megacam/ppImage.config # ppImage recipe
END

# Rejection levels for detrend creation
REJECTION METADATA
TYPE LIMITS FILTER EXPECTED IMFILE.MEAN IMFILE.STDEV EXP.MEAN EXP.STDEV EXP.MEANSTDEV ENSEMBLE.MEAN
ENSEMBLE.STDEV ENSEMBLE.MEANSTDEV IMFILE.SN EXP.SN
FLAT MULTI

BIAS LIMITS * 0 1 5 0.5 3 0.5 3 3 0 0 0
DARK LIMITS * 0 1 5 0.5 3 0.5 3 3 0 0 0
FLAT LIMITS * 0 0 0 0 0 0 0 0 3 0 0
# FLAT LIMITS u 0 0 0 0 0 0 0 0 3 0 0
# FLAT LIMITS g 0 0 0 0 0 0 0 0 3 0 0
# FLAT LIMITS r 0 0 0 0 0 0 0 0 3 0 0
# FLAT LIMITS i 0 0 0 0 0 0 0 0 3 0 0
# FLAT LIMITS z 0 0 0 0 0 0 0 0 3 0 0
FRINGE LIMITS * 0 0 0 0 0 0 0 0 0 0 0

# FILTER is an additional qualifier, and may be "*" (or absent!), in which case it matches everything
# EXPECTED is the expected mean value
# IMFILE.MEAN is the maximum permitted mean value for an imfile, relative to the standard deviation
# IMFILE.STDEV is the maximum permitted standard deviation for an imfile
# EXP.MEAN is the maximum permitted mean value for an exposure, relative to the standard deviation
# EXP.STDEV is the maximum permitted standard deviation for an exposure
# EXP.MEANSTDEV is the maximum permitted mean standard deviation for an exposure relative to the mean
# ENSEMBLE.MEAN is the maximum permitted mean for an ensemble of exposures
# ENSEMBLE.STDEV is the maximum permitted standard deviation for an ensemble of exposures
# ENSEMBLE.MEANSTDEV is the maximum permitted mean standard deviation for an ensemble of exposures
# IMFILE.SN is the minimum permitted signal-to-noise for an imfile
# EXP.SN is the minimum permitted signal-to-noise for an exposure
# These values (all except FILTER) may be zero, in which case no clipping is applied.

END


FILERULES METADATA
   ### Redirections
   PSASTRO.INPUT      STR PSASTRO.INPUT.CMP
   PSASTRO.OUTPUT     STR PSASTRO.OUTPUT.CMP
   PSPHOT.OUTPUT      STR PSPHOT.OUTPUT.CMF

   ### input file definitions
   TYPE               INPUT FILENAME.RULE                  FILENAME.XTRA EXTNAME.RULE EXTNAME.XTRA DATA.LEVEL FILE.TYP
   PPIMAGE.INPUT      INPUT @FILES                         {CHIP.NAME}   {CELL.NAME}  NONE         CHIP       IMAGE
   PPARITH.INPUT      INPUT @FILES                         {CHIP.NAME}   {CELL.NAME}  NONE         CHIP       IMAGE

   ### use these entries to get the detrend images from specific files
   PPIMAGE.MASK       INPUT mask.mef.fits                  {CHIP.NAME}   {CHIP.NAME}  NONE         CHIP       IMAGE
   PPIMAGE.BIAS       INPUT MegaCam.bias.1.0.{CHIP.NAME}.fits      {CHIP.NAME}   {CHIP.NAME}  NONE         CHIP
   PPIMAGE.DARK       INPUT MegaCam.dark.2.0.{CHIP.NAME}.fits      {CHIP.NAME}   {CHIP.NAME}  NONE         CHIP
   PPIMAGE.FLAT       INPUT MegaCam.flat.3.0.{CHIP.NAME}.fits      {CHIP.NAME}   {CHIP.NAME}  NONE         CHIP

   ### use these entries to get the detrend images from the database
   #PPIMAGE.MASK       INPUT @DETDB                        {CHIP.NAME}   {CHIP.NAME}  NONE         CHIP       IMAGE
   #PPIMAGE.BIAS       INPUT @DETDB                        fpa           fpa          NONE         CHIP       IMAGE
   #PPIMAGE.DARK       INPUT @DETDB                        {CHIP.NAME}   {CHIP.NAME}  NONE         CHIP       IMAGE
   #PPIMAGE.FLAT       INPUT @DETDB                        {CHIP.NAME}   {CHIP.NAME}  NONE         CHIP       IMAGE

   PSPHOT.INPUT       INPUT @FILES                         {CHIP.NAME}   {CELL.NAME}  NONE         CHIP       IMAGE

   PSASTRO.INPUT.CMP  INPUT @FILES                         NONE          NONE         PHU          CHIP       CMP
   PSASTRO.INPUT.CMF  INPUT @FILES                         NONE          SMPDATA      PHU          CHIP       CMF
```

```
  ### output file definitions
  TYPE                 OUTPUT FILENAME.RULE                  FILENAME.XTRA EXTNAME.RULE EXTNAME.XTRA FILE.LEVEL DATA.L
  PPIMAGE.OUTPUT       OUTPUT {OUTPUT}.{CHIP.NAME}.fits      {CHIP.NAME}   {CHIP.NAME}  NONE         CHIP       CHIP
  PPIMAGE.BIN1         OUTPUT {OUTPUT}.{CHIP.NAME}.b1.fits   {CHIP.NAME}   {CHIP.NAME}  NONE         CHIP       CHIP
  PPIMAGE.BIN2         OUTPUT {OUTPUT}.{CHIP.NAME}.b2.fits   {CHIP.NAME}   {CHIP.NAME}  NONE         CHIP       CHIP

  PPIMAGE.OUTPUT.CHIP OUTPUT {OUTPUT}.{CHIP.NAME}.chip.fits {CHIP.NAME}   {CHIP.NAME}  NONE         CHIP       CHIP
  PPIMAGE.OUTPUT.FPA1 OUTPUT {OUTPUT}.b1.fits                NONE          NONE         NONE         FPA        FPA
  PPIMAGE.OUTPUT.FPA2 OUTPUT {OUTPUT}.b2.fits                NONE          NONE         NONE         FPA        FPA

  PPIMAGE.JPEG1        OUTPUT {OUTPUT}.b1.jpg                -greyscale    RANGE        -5:20        FPA        FPA
  PPIMAGE.JPEG2        OUTPUT {OUTPUT}.b2.jpg                -greyscale    FRACTION     0.50:2.00    FPA        FPA

  PSPHOT.RESID         OUTPUT {OUTPUT}.res.fits              NONE          {CELL.NAME}  {CELL.NAME}  CHIP       CHIP
  PSPHOT.BACKGND       OUTPUT {OUTPUT}.bck.fits              NONE          {CELL.NAME}  {CELL.NAME}  CHIP       CHIP
  PSPHOT.BACKSUB       OUTPUT {OUTPUT}.sub.fits              NONE          {CELL.NAME}  {CELL.NAME}  CHIP       CHIP
  PSPHOT.BACKMDL       OUTPUT {OUTPUT}.mdl.fits              NONE          {CELL.NAME}  {CELL.NAME}  CHIP       CHIP

  PSPHOT.OUTPUT.RAW    OUTPUT {OUTPUT}.{CHIP.NAME}           NONE          NONE         PHU          CHIP       CHIP
  PSPHOT.OUTPUT.SX     OUTPUT {OUTPUT}.sx                    NONE          NONE         PHU          CHIP       CHIP
  PSPHOT.OUTPUT.OBJ    OUTPUT {OUTPUT}.obj                   NONE          NONE         PHU          CHIP       CHIP
  PSPHOT.OUTPUT.CMF    OUTPUT {OUTPUT}.cmf                   NONE          SMPDATA      PHU          CHIP       CHIP
  PSPHOT.OUTPUT.CMP    OUTPUT {OUTPUT}.{CHIP.NAME}.cmp       NONE          NONE         PHU          CHIP       CHIP

  PSASTRO.OUTPUT.CMP   OUTPUT {OUTPUT}.{CHIP.NAME}.smp       NONE          NONE         PHU          CHIP       CHIP

  PPARITH.OUTPUT       OUTPUT {OUTPUT}                       {CHIP.NAME}   {CHIP.NAME}  NONE         CHIP       CHIP

END
```

# 4 Camera format configuration

The FITS (Flexible Image Transport System) format is a standard in the astronomical community for storing astronomical images. A FITS file consists of an arbitrary number of coupled human readable `ASCII` header segments and binary data segments. The headers describe the format and layout of the data segments. The first of these groups is traditionally called the "primary header unit" (PHU) and the rest are referred to as "extensions". The header segments may contain extensive documentary information related to the interpretation of the data. Although the FITS format defines a standard representation of the data, the header metadata is not so consistently defined within the astronomical community. Also, the flexibility of the data format means that it is possible to construct a variety of different representations for the same fundamental collection of data.

The purpose of the camera format file is to define how FITS files are to be read into the Focal Plane hierarchy, and how the "concepts" are to be ingested.

## 4.1 Location

The camera formats for a particular camera are listed in the `FORMATS` metadata of the camera configuration file. Note that the `PATH` in the site configuration defines the search paths for these files.

## 4.2 Contents

The camera format specifies how a FITS file from a particular camera is to be read. Different formats may be defined for a single camera (e.g., one amplifier per extension, or all amplifiers spliced together in the PHU, or anything in between).

The camera format configuration file contains the rules for recognising the format, how to read the file, the contents of a FITS file, data appropriate to different types of cells, information on how to determine the concepts from the headers, default values, or database, and expected formats for certain concepts.

### 4.2.1 Rules for recognising

`RULE(METADATA)` contains a list of FITS headers with expected values (of the appropriate type) for this particular combination of the camera and format. It is often useful to include `TELESCOP` and `DETECTOR`, if possible, along with any other headers that uniquely identify the camera and format. Note that all of the headers must match exactly (modulo leading and trailing spaces for strings), including the data type and value, for the rule to match, and that the first format's rule to match is accepted. If a rule doesn't match the header, try adjusting the types (especially for numerical types — use S32 for integers, F32 and F64 for floats).

### 4.2.2 How to read the file

Within the FITS data representation, there are various choices which can and have been made for the placement of the pixels in the file. In the simplest case, the camera consists of a single chip consisting of a single cell always read with a single readout. In this case, the image data is generally written as part of the primary header unit. However, in a more complex case with multiple chips and multiple cells, the data may be organized in several ways. The data may be distributed into multiple files or in multiple FITS data extensions within a single file.. A single camera image may be written as a collection of files for individual chips with separate extensions for each cell (CFH12K.split, GPC). Another camera may write a single file with multiple extensions for each cell (Megacam.raw), or multiple extensions per chip, with each cell representing portions of the chip image (Megacam.splice, CFHT-IR).

In all of these representations, there are only two basic distinctions in how the pixel data is stored: what level in the hierarchy the entire FITS file corresponds to (FPA, chip, or cell), and what level the extensions correspond to (chip, cell or no extensions at all). Knowing these, and having a list of the contents of each extension, we can construct the Focal Plane hierarchy.

`FILE(METADATA)` contains information on how to read the FITS file for this format. The contents are:

- `PHU(STR)` identifies the class of the file — what level in the focal plane hierarchy the primary header unit (PHU) of this file belongs. Legal values are `FPA`, `CHIP` or `CELL`.
- `EXTENSIONS(STR)` identifies what level in the focal plane hierarchy the extensions belong. Legal values are `CHIP`, `CELL` or `NONE` (if there are no extensions).
- `FPA.NAME(STR)` specifies a PHU header keyword for a unique identifier for the FPA. This is usually an exposure number, or similar. The purpose is to identify the FPA, so that only files with the same value of `FPA.NAME` can be admitted to the same FPA structure.
- `CHIP.NAME(STR)` (only required if `PHU` is `CHIP` or `CELL`) specifies a PHU header keyword that identifies the name of the chip. The purpose is to identify to which chip in the hierarchy the file belongs.
- `CELL.NAME(STR)` (only required if `PHU` is `CELL`) specifies a PHU header keyword that identifies the name of the cell within the chip. The purpose is to identify to which cell in the hierarchy the file belongs.

- `CONTENT(STR)` (only required if `EXTENSIONS` is `NONE` and `PHU` is `CHIP` or `CELL`) specifies a key to the `CONTENTS` menu (see below). The purpose is to identify the contents of the file (in terms of its FPA hierarchy components). The string has concepts interpolated, where these are enclosed in curly brackets (currently `CHIP.NAME` and `CELL.NAME` only; **future concepts may be permitted in the future if there exists sufficient demand (TBD)** . This allows such a construct as `{CHIP.NAME}_{CELL.NAME}` to identify a combination of chip and cell.

#### 4.2.3 File contents

The exact meaning of the `CONTENTS` (as well as the type) depends on the value of `PHU` and `EXTENSIONS` in the `FILE` metadata. In each case, we rely on the use of `chip:cell:type` triplets to identify the contents. These are used to identify the contents of an extension: the chip and cell to which a component belongs, and the type of the cell (see §4.2.4 for cell types), with the symbolic names separated by colons. Where an extension contains more than one cell, the triplets are listed one after the other, separated by whitespace.

- If `PHU` is `FPA` and `EXTENSIONS` is `NONE`, then `CONTENTS` is of type `STR`, and contains a string of `chip:cell:type` triplets.
- If `PHU` is `CHIP` or `CELL` and `EXTENSIONS` is `NONE`, then `CONTENTS` is of type `METADATA`, and contains a menu of possible contents. Each menu item is of type `STR`, and consists of a string of `chip:cell:type` triplets. The menu key is provided by the interpolated `CONTENT` value within the `FILE` metadata.
- In all other cases, `CONTENTS` is of type `METADATA`, and contains a list of extension names within the file, with the values of type `STR` consisting of a string of `chip:cell:type` triplets.

#### 4.2.4 Cell data

`CELLS(METADATA)` contains a list of cell types, with concepts particular to those types. Each type, which corresponds to a type specified in the `CONTENTS`, is of type `METADATA`. The contents of these metadata are values for concepts that are particular to that cell type (e.g., left amplifier vs right amplifier). Usually `CELL.TRIMSEC(STR)` and `CELL.BIASSEC(STR)` will be listed here, since these differ according to the cell type. Since there is ambiguity in what the values here refer to (if the concept is of type `STR`, then the value could be a header name or the actual value to use), we also require an additional entry with `.SOURCE` suffixed to the concept name, with the value (of type `STR`) being `VALUE` to indicate that the concept is specified by value, or `HEADER` to indicate that the concept is specified in the header of the given name.

[It might be thought that there is no need to provide the ability to look up headers here, since it is provided below. However, the header name may vary depending on the cell type. For example, the Megacam spliced format uses `TSECA` and `TSECB` to specify the trim sections for the left and right amplifiers, respectively.]

#### 4.2.5 Concepts from headers

`TRANSLATION(METADATA)` contains a list of concepts that have their values ingested from the FITS headers. Each concept name should have type `STR`, with the value being the header name from which the concept is ingested. No distinction is made between the PHU and extension headers, but inheritance (look at the PHU if it's not in the extension header) should be the normal behaviour. Multiple header keywords (separated by whitespace) may be given for certain concepts:

- FPA.TIME and CELL.TIME to specify the date and time (in that order) are contained in separate header keywords.
- CELL.BIASSEC to specify multiple bias regions (e.g., a prescan and an overscan).

**TRANSLATION is a poor name (it's supposed to be a header translation table); HEADERS would be better. (TBD)**

#### 4.2.6 Concepts from default values

DEFAULTS(METADATA) contains a list of concepts with their default values (of the appropriate types). A concept may have type METADATA, in which case the metadata acts as a menu. The menu key is determined from an additional entry in the DEFAULTS, formed from the concept name suffixed with .DEPEND, which must be of type STR and contain a concept name. The value of this extra concept determines the menu key. This allows dependence on the chip (e.g., depending on CHIP.NAME) or cell (CELL.NAME), which is useful for setting things such as CHIP.X0 when it is not contained in the header.

#### 4.2.7 Concepts from database

**Database lookup for concepts has never been tested. In fact, the current implementation probably doesn't even match this description. (TBD)**

DATABASE(METADATA) contains a list of concepts whose values are determined from database lookup. Each concept is of type METADATA. Each concept metadata must contain the entries TABLE(STR) and COLUMN(STR), which specify the database table to use, and the column within that table. Additional entries provide the WHERE part of the database query.

#### 4.2.8 Formats for concepts

FORMATS(METADATA) contains a list of concepts that require additional information in order to parse. Each concept name contains a value of type STR which is a list of options for parsing the concept.

Concepts which require formats:

- FPA.RA and FPA.DEC: the format specifies the units — HOURS, DEGREES or RADIANS. FPA.RA defaults to HOURS, and FPA.DEC defaults to DEGREES.
- FPA.TIME and CELL.TIME: USA indicates that the date format is mm-dd-yyyy; BACKWARDS indicates that the date format is dd-mm-yyyy; PRE2000 indicates that a two-digit date is used (1900 years is added if the year is less than 100); MJD indicates the date is a modified julian date; JD indicates the date is a julian date.
- CELL.X0, CELL.Y0, CHIP.X0 and CHIP.Y0: FORTRAN indicates that the corner lower left-hand pixel corresponds to coordinates (1,1); if missing, assumes that the corner is at (0,0).

#### 4.2.9 Default concepts

Default concepts that should be included in each camera format file, either in the `CELLS`, `TRANSLATION`, `DEFAULTS` or `DATABASE`:

- `FPA.TELESCOPE`: Telescope used
- `FPA.INSTRUMENT`: Instrument used
- `FPA.DETECTOR`: Detector used
- `FPA.CAMERA`: Camera used; **To be deprecated? (TBD)**
- `FPA.FOCUS`: Telescope focus
- `FPA.AIRMASS`: Airmass at boresight
- `FPA.FILTER`: Filter used
- `FPA.FILTERID`: Filter identifier (parsed through the `FILTER.ID` translation table in the camera configuration).
- `FPA.POSANGLE`: Position angle of instrument
- `FPA.RADECSYS`: Celestial coordinate system
- `FPA.RA`: Right Ascension of boresight
- `FPA.DEC`: Declination of boresight
- `FPA.OBSTYPE`: Type of observation
- `FPA.OBJECT`: Object of observation
- `FPA.ALT`: Altitude of telescope
- `FPA.AZ`: Azimuth of telescope
- `FPA.TIMESYS`: Time system
- `FPA.TIME`: Time of exposure
- `FPA.TEMP`: Temperature of the focal plane
- `FPA.EXPOSURE`: Exposure time for the focal plane
- `CHIP.XPARITY`: Orientation in x compared to the rest of the FPA
- `CHIP.YPARITY`: Orientation in y compared to the rest of the FPA
- `CHIP.X0`: Position of (0,0) on the FPA
- `CHIP.Y0`: Position of (0,0) on the FPA
- `CHIP.TEMP`: Temperature of chip

- CELL.GAIN: CCD gain (e/count)
- CELL.READNOISE: CCD read noise (e)
- CELL.SATURATION: Saturation level (counts)
- CELL.BAD: Bad level (counts)
- CELL.XPARITY: Orientation in x compared to the rest of the chip
- CELL.YPARITY: Orientation in y compared to the rest of the chip
- CELL.READDIR: Read direction, rows=1, cols=2
- CELL.EXPOSURE: Exposure time (sec)
- CELL.DARKTIME: Time since flush (sec)
- CELL.TRIMSEC: Trim section
- CELL.BIASSEC: Bias sections
- CELL.XBIN: Binning in x
- CELL.YBIN: Binning in y
- CELL.TIMESYS: Time system
- CELL.TIME: Time of exposure
- CELL.X0: Position of (0,0) on the chip
- CELL.Y0: Position of (0,0) on the chip

In addition, FPA.NAME, CHIP.NAME and CELL.NAME are included automatically, based on the FILE and CONTENTS metadatas.

## 4.3 Examples

### 4.3.1 Megacam (short) raw

```
# "mcshort" is a MegaCam camera with only the central six chips --- it’s faster than the entire FPA.
# The raw MegaCam data comes off the telescope with each of the chips stored in extensions of a MEF file.

# How to identify this type
RULE    METADATA
        TELESCOP        STR     CFHT 3.6m
        DETECTOR        STR     MegaCam
        EXTEND          BOOL    T
        NEXTEND         S32     72
END

# How to read this data
FILE    METADATA
        PHU             STR     FPA     # The FITS file represents an entire FPA
        EXTENSIONS      STR     CELL    # The extensions represent cells
        FPA.NAME        STR     EXPNUM  # A PHU keyword for unique identifier within the hierarchy level
```

```
END

# What's in the FITS file?
CONTENTS        METADATA
        # Extension name, chip:cell:type
        amp24           STR     ccd12:LeftAmp:left
        amp25           STR     ccd12:RightAmp:right
        amp26           STR     ccd13:LeftAmp:left
        amp27           STR     ccd13:RightAmp:right
        amp28           STR     ccd14:LeftAmp:left
        amp29           STR     ccd14:RightAmp:right
        amp42           STR     ccd21:LeftAmp:left
        amp43           STR     ccd21:RightAmp:right
        amp44           STR     ccd22:LeftAmp:left
        amp45           STR     ccd22:RightAmp:right
        amp46           STR     ccd23:LeftAmp:left
        amp47           STR     ccd23:RightAmp:right
END

# Specify the cell data
CELLS   METADATA
        left    METADATA        # Left amplifier
                CELL.BIASSEC.SOURCE     STR     HEADER
                CELL.TRIMSEC.SOURCE     STR     HEADER
                CELL.BIASSEC            STR     BIASSEC
                CELL.TRIMSEC            STR     DATASEC
                CELL.XPARITY            S32     1 # We could have specified this as a DEFAULT, but this works
                CELL.X0                 S32     1
        END
        right   METADATA        # Right amplifier
                CELL.BIASSEC.SOURCE     STR     HEADER
                CELL.TRIMSEC.SOURCE     STR     HEADER
                CELL.BIASSEC            STR     BIASSEC
                CELL.TRIMSEC            STR     DATASEC
                CELL.XPARITY            S32     -1 # This cell is read out in the opposite direction
                CELL.X0                 S32     2048
        END
END

# How to translate PS concepts into FITS headers
TRANSLATION     METADATA
        FPA.NAME                STR     EXPNUM
        FPA.AIRMASS             STR     AIRMASS
        FPA.FILTER              STR     FILTER
        FPA.POSANGLE            STR     ROTANGLE
        FPA.RA                  STR     RA
        FPA.DEC                 STR     DEC
        FPA.RADECSYS            STR     RADECSYS
        FPA.OBSTYPE             STR     OBSTYPE
        FPA.OBJECT              STR     CMMTOBS
        FPA.TIME                STR     MJD-OBS
        FPA.TIMESYS             STR     TIMESYS
        FPA.ALT                 STR     TELALT
        FPA.AZ                  STR     TELAZ
        CHIP.TEMP               STR     DETTEM
        CELL.EXPOSURE           STR     EXPTIME
        CELL.DARKTIME           STR     DARKTIME
        CELL.GAIN               STR     GAIN
        CELL.READNOISE          STR     RDNOISE
        CELL.SATURATION         STR     SATURATE
        CELL.TIME               STR     MJD-OBS
        CELL.TIMESYS            STR     TIMESYS
        CELL.XBIN               STR     CCDBIN1
        CELL.YBIN               STR     CCDBIN2
END

# Default PS concepts that may be specified by value
DEFAULTS        METADATA
```

```
        CELL.READDIR            S32     1               # Cell is read in x direction
        CELL.BAD                S32     0
        CELL.YPARITY            S32     1
        CELL.Y0                 S32     1

        CHIP.X0.DEPEND          STR     CHIP.NAME
        CHIP.X0         METADATA
                ccd12   S32     6144
                ccd13   S32     8192
                ccd14   S32     10240
                ccd21   S32     6144
                ccd22   S32     8192
                ccd23   S32     10240
        END
        CHIP.Y0.DEPEND          STR     CHIP.NAME
        CHIP.Y0         METADATA
                ccd12   S32     13835
                ccd13   S32     13835
                ccd14   S32     13835
                ccd21   S32     4612
                ccd22   S32     4612
                ccd23   S32     4612
        END
        CHIP.XPARITY.DEPEND     STR     CHIP.NAME
        CHIP.XPARITY    METADATA
                ccd12   S32     1
                ccd13   S32     1
                ccd14   S32     1
                ccd21   S32     1
                ccd22   S32     1
                ccd23   S32     1
        END
        CHIP.YPARITY.DEPEND     STR     CHIP.NAME
        CHIP.YPARITY    METADATA
                ccd12   S32     -1
                ccd13   S32     -1
                ccd14   S32     -1
                ccd21   S32     1
                ccd22   S32     1
                ccd23   S32     1
        END
END

# How to translate PS concepts into database lookups
DATABASE        METADATA
        TYPE            dbLookup        TABLE           COLUMN          chipId          cellId
#       CHIP.TEMP       METADATA
#               TABLE   STR     Cryostat
#               COLUMN  STR     temp
#               chipId  STR     {CHIP.NAME}
#               time    STR     {CELL.TIME}
#       END
#       CELL.GAIN       dbLookup        Camera          gain            CHIP.NAME       CELL.NAME
#       CELL.READNOISE  dbLookup        Camera          readNoise       CHIP.NAME       CELL.NAME
END


# Where there might be some ambiguity, specify the format
FORMATS         METADATA
        FPA.RA          STR     HOURS
        FPA.DEC         STR     DEGREES
        FPA.TIME        STR     MJD
        CELL.TIME       STR     MJD
        CELL.X0         STR     FORTRAN
        CELL.Y0         STR     FORTRAN
END
```

### 4.3.2 Megacam (short) split

```
# "mcshort" is a MegaCam camera with only the central six chips --- it’s faster than the entire FPA.
# The spliced MecaCam data is stored in single extensions for each chip

# How to recognise this type
RULE    METADATA
        TELESCOP        STR     CFHT 3.6m
        DETECTOR        STR     MegaCam
        # No particular distinguishing features apart from these, so we list this format last
        # in the camera configuration file.
END

FILE    METADATA
        # How to read this data
        PHU             STR     CHIP    # The FITS file represents an entire FPA
        EXTENSIONS      STR     NONE    # The extensions represent chips
        FPA.NAME        STR     EXPNUM  # A PHU keyword for unique identifier
        CHIP.NAME       STR     EXTNAME # An extension keyword for unique identifie
        CONTENT         STR     {CHIP.NAME} # Key to the CONTENTS menu
END

# What’s in the FITS file?
CONTENTS        METADATA
        # Extension name, chip:cell:type
        ccd12           STR     ccd12:LeftAmp:left ccd12:RightAmp:right
        ccd13           STR     ccd13:LeftAmp:left ccd13:RightAmp:right
        ccd14           STR     ccd14:LeftAmp:left ccd14:RightAmp:right
        ccd21           STR     ccd21:LeftAmp:left ccd21:RightAmp:right
        ccd22           STR     ccd22:LeftAmp:left ccd22:RightAmp:right
        ccd23           STR     ccd23:LeftAmp:left ccd23:RightAmp:right
END

# Specify the cells
CELLS           METADATA
        left            METADATA
                CELL.BIASSEC.SOURCE     STR     HEADER
                CELL.TRIMSEC.SOURCE     STR     HEADER
                CELL.BIASSEC            STR     BSECA
                CELL.TRIMSEC            STR     TSECA
                CELL.X0                 S32     0
                CELL.GAIN.SOURCE        STR     HEADER
                CELL.GAIN               STR     GAINA
        END

        right           METADATA
                CELL.BIASSEC.SOURCE     STR     HEADER
                CELL.TRIMSEC.SOURCE     STR     HEADER
                CELL.BIASSEC            STR     BSECB
                CELL.TRIMSEC            STR     TSECB
                CELL.X0                 S32     1024
                CELL.GAIN.SOURCE        STR     HEADER
                CELL.GAIN               STR     GAINB
        END
END

# How to translate PS concepts into FITS headers
TRANSLATION     METADATA
        FPA.NAME        STR     EXPNUM
        FPA.AIRMASS     STR     AIRMASS
        FPA.FILTER      STR     FILTER
        FPA.POSANGLE    STR     ROTANGLE
        FPA.RA          STR     RA
        FPA.DEC         STR     DEC
        FPA.RADECSYS    STR     RADECSYS
        FPA.OBSTYPE     STR     OBSTYPE
        FPA.OBJECT      STR     CMMTOBS
```

```
        FPA.TIME        STR     MJD-OBS
        FPA.TIMESYS     STR     TIMESYS
        FPA.ALT         STR     TELALT
        FPA.AZ          STR     TELAZ
        CHIP.TEMP       STR     DETTEM
        CELL.EXPOSURE   STR     EXPTIME
        CELL.DARKTIME   STR     DARKTIME
        CELL.READNOISE  STR     RDNOISE
        CELL.SATURATION STR     SATURATE
        CELL.TIME       STR     MJD-OBS
        CELL.TIMESYS    STR     TIMESYS
        CELL.XBIN       STR     CCDBIN1
        CELL.YBIN       STR     CCDBIN2
END

# Default PS concepts that may be specified by value
DEFAULTS        METADATA
        CELL.READDIR            S32     1               # Cell is read in x direction
        CELL.BAD                S32     0
        CELL.XPARITY            S32     1
        CELL.YPARITY            S32     1
        CELL.Y0                 S32     0
#       PPMERGE.SCALE           F32     1.0
#       PPMERGE.ZERO            F32     0.0
        CHIP.X0.DEPEND          STR     CHIP.NAME
        CHIP.X0         METADATA
                ccd12   S32     0
                ccd13   S32     2048
                ccd14   S32     4096
                ccd21   S32     0
                ccd22   S32     2048
                ccd23   S32     4096
        END
        CHIP.Y0.DEPEND          STR     CHIP.NAME
        CHIP.Y0         METADATA
                ccd12   S32     9223
                ccd13   S32     9223
                ccd14   S32     9223
                ccd21   S32     0
                ccd22   S32     0
                ccd23   S32     0
        END
        CHIP.XPARITY.DEPEND     STR     CHIP.NAME
        CHIP.XPARITY    METADATA
                ccd12   S32     1
                ccd13   S32     1
                ccd14   S32     1
                ccd21   S32     1
                ccd22   S32     1
                ccd23   S32     1
        END
        CHIP.YPARITY.DEPEND     STR     CHIP.NAME
        CHIP.YPARITY    METADATA
                ccd12   S32     -1
                ccd13   S32     -1
                ccd14   S32     -1
                ccd21   S32     1
                ccd22   S32     1
                ccd23   S32     1
        END
END


# How to translation PS concepts into database lookups
DATABASE        METADATA
# None
END
```

```
# Where there might be some ambiguity, specify the format
FORMATS         METADATA
        FPA.RA          STR     HOURS
        FPA.DEC         STR     DEGREES
        FPA.TIME        STR     MJD
        CELL.TIME       STR     MJD
END
```

#### 4.3.3 Imaging Sky Probe

```
# Pan-STARRS Imaging Sky Probe

# How to identify this type
RULE    METADATA
        SIMPLE          BOOL    TRUE
        NAXIS           S32     2
        TELESCOP        STR     ISP-1
        INSTRUME        STR     ISP-Apogee
        DETECTOR        STR     ISP-Apogee-01
        ISPCAMER        STR     Apogee U42
END

# How to read this data
FILE    METADATA
        PHU             STR     FPA     # The FITS file represents an entire FPA
        EXTENSIONS      STR     NONE    # There are no extensions
        FPA.NAME        STR     SEQID   # A PHU keyword for unique identifier within the hierarchy level
END

# What's in the FITS file?
CONTENTS        STR     Chip:Cell:amplifier

# Specify the cell data
CELLS   METADATA
        amplifier       METADATA
                CELL.TRIMSEC.SOURCE     STR     HEADER
                CELL.BIASSEC.SOURCE     STR     HEADER
                CELL.TRIMSEC            STR     TRIMSEC
                CELL.BIASSEC            STR     BIASSEC
        END
END

# How to translate PS concepts into FITS headers
TRANSLATION     METADATA
        FPA.OBSTYPE     STR     OBSTYPE
        FPA.OBJECT      STR     OBSTYPE
        FPA.FILTER      STR     FILTNAME
        FPA.RA          STR     RA
        FPA.DEC         STR     DEC
        FPA.RADECSYS    STR     RADECSYS
        FPA.ALT         STR     ALT
        FPA.AZ          STR     AZ
        FPA.POSANGLE    STR     ROTANGLE
        FPA.AIRMASS     STR     AIRMASS
        FPA.TIME        STR     MJD-OBS
        CHIP.TEMP       STR     CCDTEMP
        CELL.EXPOSURE   STR     EXPTIME
        CELL.DARKTIME   STR     DARKTIME
        CELL.TIME       STR     MJD-OBS
        CELL.GAIN       STR     GAIN
        CELL.READNOISE  STR     RDNOISE
        CELL.XBIN       STR     XBIN
        CELL.YBIN       STR     YBIN
#       CELL.SATURATION STR     SATURATE        ### Currently set to 0 ???
```

```
        CELL.BAD        STR     BADLEVEL
END

# Default PS concepts that may be specified by value
DEFAULTS        METADATA
        FPA.TIMESYS     STR     UTC
        CELL.SATURATION F32     65535
        CELL.READDIR    S32     1
        CELL.TIMESYS    STR     UTC
        CHIP.XPARITY    S32     1
        CHIP.YPARITY    S32     1
        CHIP.X0         S32     0
        CHIP.Y0         S32     0
        CELL.XPARITY    S32     1
        CELL.YPARITY    S32     1
        CELL.X0         S32     0
        CELL.Y0         S32     0
END

FORMATS         METADATA
        FPA.RA          STR     HOURS
        FPA.DEC         STR     DEGREES
        FPA.TIME        STR     MJD
        CELL.TIME       STR     MJD
END

# PS Concepts to get from the database
DATABASE        METADATA
# None.
END
```

# 5 Recipes

## 5.1 Locations

Recipes may be specified in a number of locations. The recipe files are loaded in a sequence, with each new file supplementing the recipes already defined. First, the site-wide list of recipes is loaded. Next, if a camera can be identified, the camera-specific recipes are loaded. In both locations, the recipes are identified as named files under the `RECIPES` metadata. Note that the `PATH(STR)` in the site configuration defines the search paths for these files. Finally, they may be specified on the command line with the `-recipe` option, giving a symbolic name and a filename or another symbolic name to link to. In addition, individual recipe values may be specified on the command line with one of several command-line options.

### 5.1.1 Recipe combination

A single recipes are defined at multiple levels (site, camera, and command-line), so it's important to know how these are loaded. The site configuration recipes serve as the default recipes. Once the particular camera is known, the values contained within its recipes (provided either as a filename or as a symbolic link; see below for symbolic links) override those defined in the site configuration (unless the value has been declared as `MULTI`, in which case it supplements). This is useful because recipes often depend on the camera from which the data being processed originated; for example, not all cameras require a dark to be subtracted.

Finally, the command line can be used to provide further refinement. A recipe can be defined on the command line using `-recipe RECIPE_NAME filename.config` to specify a file containing the recipe, or `-recipe RECIPE_NAME`

`ALTERNATE_RECIPE_NAME` to specify an symbolic link from which to inget values for the original recipe.

Symbolic links offer the ability to override the default recipe values by specifying a name, rather than a filename. A symbolic link can refer to a recipe of a different name that has already been defined, or it can refer to a `METADATA` within the recipe of that same name.

A few examples are useful here. Say the site configuration contains:

```
RECIPES    METADATA
    RECIPE         STR    recipe_default.config
    RECIPE_EXOTIC  STR    recipe_exotic.config
END
```

The camera configuration has:

```
RECIPES    METADATA
    RECIPE         STR    recipe_camera.config
END
```

`recipe_default.config` has:

```
VALUE    STR    Default

RECIPE_DULL    METADATA
    VALUE    STR    Dull
END
```

`recipe_exotic.config` has:

```
VALUE    STR    Exotic
```

And `recipe_camera.config` has:

```
VALUE    STR    Camera
```

Then:

- If the recipe is examined without knowing the camera, `VALUE` will be `Default`.
- If the recipe is examined once the camera is known, `VALUE` will be `Camera`.
- If the command-line contains `-recipe RECIPE recipe_exotic.config`, `VALUE` will be `Exotic`.
- If the command-line contains `-recipe RECIPE RECIPE_EXOTIC`, `VALUE` will be `Exotic`.
- If the command-line contains `-recipe RECIPE RECIPE_DULL`, `VALUE` will be `Dull`.

The priority for recipe sources is:

5.1.1.1 Site configuration

5.1.1.2 Camera configuration

#### 5.1.1.3 Command-line recipes

#### 5.1.1.4 Command-line options

If multiple recipes have the same name, higher priority entries over-write the values specified in the lower-priority entries. Values which are not defined in the higher-priority entries are inherited from the lower-priority entries. This allows the user to override any recipe values using the command-line, and to specify default values in the site configuration, while also having camera-specific values in the camera configurations. A good practice is for the higher-priority recipes files to only supply the entries which are different from the default values.

```
loading sequence is:

* in pmConfigRead->pmConfigReadRecipes(config, PM_RECIPE_SOURCE_SITE | PM_RECIPE_SOURCE_CAMERA):

  config->site:RECIPES:NAME=FILE(STR) -> config->recipes:NAME(MD)

  config->camera:RECIPES:NAME=FILE(STR) -> config->recipes:NAME(MD)
  (if camera is specified)

* in pmConfigRead->pmConfigLoadRecipeArguments (config):

  config->argv:-recipe:NAME=FILE(STR) -> config->arguments:RECIPES:NAME(MD)

  config->argv:-recipe:NAME=REF(STR) -> config->recipesSource:NAME=REF(STR)

* in pmConfigRead->pmConfigLoadRecipeOptions (config, "-D"):

  config->argv:-D:NAME:KEY=VALUE(STR) -> config->arguments:OPTIONS:NAME(MD)

** file is loaded / camera is identified **

* in pmConfigCameraFormatFromHeader->pmConfigReadRecipes(config, PM_RECIPE_SOURCE_CAMERA | PM_RECIPE_SOURCE_CL):

  config->camera:RECIPES:NAME=FILE(STR) -> config->recipes:NAME(MD)

** at this point, all recipes are loaded as MD in either config->recipes, config->arguments:RECIPES, or config->argum

  config->arguments:RECIPES:NAME(MD) -> config->recipes:NAME(MD)

** resolve the symbolic names:

  config->recipesSource:NAME=REF(STR):
  * if REF is in config->recipes, interpolate: config->recipes:REF(MD) -> config->recipes:NAME(MD)
  * else if REF is in config->recipes:NAME, interpolate: config->recipes:NAME:REF(MD) -> config->recipes:NAME(MD)

** apply the OPTIONS:

  config->arguments:OPTIONS:NAME(MD) -> config->recipes:NAME(MD)

--

Examples:
program -recipe PPIMAGE ppImage.config -recipe PPIMAGE PPIMAGE_BIAS
```

## 5.2 Contents

The contents of the recipe files depends on the particular recipe.

### 5.2.1 PPIMAGE

The `PPIMAGE` recipe contains options for `ppImage`:

- `MASK(BOOL)` indicates if bad pixels are to be masked.
- `MASK.VALUE(U8)` specifies a bitmask (matching the bad pixel mask) for pixels to mask in the input image.
- `NONLIN(BOOL)` indicates if the non-linearity correction is to be performed.
- `OVERSCAN(BOOL)` indicates if the overscan correction is to be performed.
- `BIAS(BOOL)` indicates if the bias correction is to be performed.
- `DARK(BOOL)` indicates if the dark correction is to be performed.
- `SHUTTER(BOOL)` indicates if the shutter correction is to be performed.
- `FLAT(BOOL)` indicates if the flat-field correction is to be performed.
- `FRINGE(BOOL)` indicates if the fringe correction is to be performed.
- `PHOTOM(BOOL)` indicates if the photometry is to be performed.
- `ASTROM.CHIP(BOOL)` indicates if the astrometry is to be performed on a chip level.
- `ASTROM.MOSAIC(BOOL)` indicates if the astrometry is to be performed on a mosaic (FPA) level.
- `BASE.FITS(BOOL)` indicates if the base detrended image is to be saved.
- `CHIP.FITS(BOOL)` indicates if the chip mosaicked image is to be saved.
- `FPA1.FITS(BOOL)` indicates if the FPA mosaicked image with first level binning is to be saved.
- `FPA2.FITS(BOOL)` indicates if the FPA mosaicked image with second level binning is to be saved.
- `BIN1.FITS(BOOL)` indicates if the chip mosaicked image with first level binning is to be saved.
- `BIN2.FITS(BOOL)` indicates if the chip mosaicked image with second level binning is to be saved.
- `BIN1.JPEG(BOOL)` indicates if the JPEG image with first level binning is to be saved.
- `BIN2.JPEG(BOOL)` indicates if the JPEG image with second level binning is to be saved.
- `NONLIN.DATA` may be:
  - A vector of type `F32`, in which case it provides the (ordinary) polynomial coefficients for the non-linear correction.
  - Of type `STR`, in which case it provides a filename with the lookup table (consisting of two columns of values, the first the input flux and the second the corresponding corrected flux).
  - Of type `METADATA`, in which case it is a menu, with menu items with types and values according to one of the other two options. The menu key is provided by `NONLIN.SOURCE(STR)`, which gives a concept name to look up (`CHIP.NAME` would be a good choice).

**Non-linearity correction is implemented but not tested. (TBD)**

- OVERSCAN.SINGLE(BOOL) indicates if the entire overscan is to be reduced to a single value.
- OVERSCAN.FIT(STR) indicates the type of fit that is to be performed to the overscan (if OVERSCAN.SINGLE is FALSE): NONE, POLYNOMIAL or SPLINE.
- OVERSCAN.ORDER(S32) gives the order of the polynomial fit (or number of spline pieces).
- OVERSCAN.STAT(STR) gives the statistic to apply to the overscan: MEAN or MEDIAN. **Would like to change this to allow the full range of statistics. (TBD)**
- FRINGE.ITER(S32) specifies the number of rejection iterations for fringe solution.
- FRINGE.REJ(F32) specifies the rejection threshold (in standard deviations) for fringe solution.
- FRINGE.KEEP(F32) specifies the minimum fraction of points to keep in the fringe solution.
- BIN1.XBIN(S32) gives the level 1 binning in x
- BIN2.YBIN(S32) gives the level 1 binning in y
- BIN2.XBIN(S32) gives the level 2 binning in x
- BIN2.YBIN(S32) gives the level 2 binning in y:
- PHOTCODE.RULE(STR) gives a rule for producing a photometry code, with values in curly brackets interpolated in the same manner as the file rules in the camera configuration.
- PPIMAGE.JPEG1(METADATA) and PPIMAGE.JPEG2(METADATA) give parameters for JPEG scaling, and contains:
  - COLORMAP(STR) specifies the colormap to use: greyscale, -greyscale (inverse greyscale), rainbow, heat.
  - SCALE.MODE(STR) specifies how the scaling is performed: RANGE or FRACTION.
  - SCALE.MIN(F32) specifies the minimum scale.
  - SCALE.MAX(F32) specifies the maximum scale.
- DETREND.CONSTRAINTS(METADATA) contains constraints for the detrend lookup as a function of type. Each type is a METADATA with the constraints to be used. Supported constraint names are: FILTER, EXPTIME and AIRMASS. Each is of type STR and the value is a string specifying a concept that will provide the constraint value.

#### 5.2.1.1 Example

```
### ppImage recipe configuration file

# List of tasks to perform
MASK            BOOL    FALSE           # Mask bad pixels
MASK.VALUE      U8      0xff            # Only mask pixels matching this bitmask
NONLIN          BOOL    FALSE           # Non-linearity correction
OVERSCAN        BOOL    TRUE            # Overscan subtraction
BIAS            BOOL    TRUE            # Bias subtraction
DARK            BOOL    TRUE            # Dark subtraction
FLAT            BOOL    TRUE            # Flat-field normalisation
FRINGE          BOOL    FALSE           # Fringe subtraction
PHOTOM          BOOL    FALSE           # Source identification and photometry
```

```
ASTROM.CHIP     BOOL    FALSE           # Astrometry on chip
ASTROM.MOSAIC   BOOL    FALSE           # Astrometry on mosaic

BASE.FITS       BOOL    TRUE            # Save base detrended image?
CHIP.FITS       BOOL    TRUE            # Save chip-mosaic-ed image?
FPA1.FITS       BOOL    TRUE            # Save 1st binned fpa image?
FPA2.FITS       BOOL    TRUE            # Save 2nd binned fpa image?
BIN1.FITS       BOOL    TRUE            # Save 1st binned chip image?
BIN2.FITS       BOOL    TRUE            # Save 2nd binned chip image?
BIN1.JPEG       BOOL    TRUE            # Save 1st binned jpeg?
BIN2.JPEG       BOOL    FALSE           # Save 2nd binned jpeg?

# Non-linearity correction
NONLIN.SOURCE           STR     CHIP.NAME       # How to determine the source
#@NONLIN.DATA           F32     0.0 1.001 0.001 # A polynomial
#NONLIN.DATA            STR     nonlin.dat      # Filename for lookup table
NONLIN.DATA             METADATA                # Source of non-linearity data
        ccd00           STR     nonlin00.dat    # A lookup table
        @ccd01          F32     0.0 1.001 0.001 # A polynomial
        @ccd02          F32     1.2345          # A polynomial
END

# Overscan subtraction
OVERSCAN.SINGLE         BOOL    FALSE           # Reduce overscan to a single value?
#OVERSCAN.FIT           STR     SPLINE          # NONE | POLYNOMIAL | SPLINE
OVERSCAN.FIT            STR     POLYNOMIAL      # NONE | POLYNOMIAL | SPLINE
OVERSCAN.ORDER          S32     5               # Order of polynomial fit
OVERSCAN.STAT           STR     MEAN            # MEAN | MEDIAN

# Fringe subtraction options
FRINGE.ITER S32 10 # Number of rejection iterations for fringe solution
FRINGE.REJ F32 2.0 # Rejection threshold for fringe solution
FRINGE.KEEP F32 0.5 # Minimum fraction to keep in fringe solution

# binned output image options
BIN1.XBIN               S32     8
BIN1.YBIN               S32     8
BIN2.XBIN               S32     64
BIN2.YBIN               S32     64

PPIMAGE.JPEG1  METADATA
  COLORMAP      STR     -greyscale
  SCALE.MODE    STR     RANGE
  SCALE.MIN     F32     -5.0
  SCALE.MAX     F32     20.0
END

PPIMAGE.JPEG2  METADATA
  COLORMAP      STR     greyscale
  SCALE.MODE    STR     FRACTION
  SCALE.MIN     STR     0.50
  SCALE.MAX     STR     2.00
END

PHOTCODE.RULE           STR     {CAMERA}.{FILTER.ID}.{CHIP.N}

DETREND.CONSTRAINTS  METADATA
  BIAS METADATA
  # NONE
  END
  DARK METADATA
    EXPTIME STR FPA.EXPTIME
  END
  FLAT METADATA
    FILTER  STR FPA.FILTERID
  END
  FRINGE METADATA
    FILTER   STR FPA.FILTERID
```

```
    AIRMASS  STR FPA.AIRMASS
  END
  SHUTTER METADATA
  # NONE
  END
END
```

### 5.2.2 PPMERGE

The PPMERGE recipe contains options for ppMerge:

- ROWS(S32) gives the number of rows to be read at once (a number larger than the physical size will read all rows).
- ELECTRONS(F32) gives the minimum number of electrons for useful signal. **Don't think this is implemented yet. (TBD)**
- SAMPLE(S32) specifies a sampling frequency for determining the background level.
- REJ(F32) specifies a rejection threshold, in standard deviations.
- ITER(S32) specifies the number of rejection iterations.
- FRACHIGH(F32) gives the fraction of high pixels to reject immediately.
- FRACLOW(F32) gives the fraction of low pixels to reject immediately.
- NKEEP(S32) gives the minimum number of pixels in the stack to keep.
- MASKVAL(S32) gives the mask value for input data.
- COMBINE(STR) gives the statistic to use for combination.
- MEAN(STR) gives the statistic to use to measure the mean.
- STDEV(STR) gives the statistic to use to measure the standard deviation.
- WEIGHTS(BOOL) specifies whether image (Poisson) weights should be used in the combination.
- For combining a fringe:
  - FRINGE.NUM(S32) specifies the number of fringe regions for fringe combination.
  - FRINGE.SIZE(S32) specifies the half-size of the fringe regions.
  - FRINGE.XSMOOTH(S32) specifies the number of smoothing regions in x.
  - FRINGE.YSMOOTH(S32) specifies the number of smoothing regions in y.
- For generating a shutter correction:
  - SHUTTER.SIZE(S32) specifies the size for shutter measurement regions.
  - SHUTTER.ITER(S32) specifies the number of iterations for performing the shutter measurement.
  - SHUTTER.REJECT(F32) specifies the rejection limit for shutter measurement.
- For generating a bad pixel mask:

  - `MASK.SUSPECT(F32)` specifies the threshold for suspect pixels (in standard deviations).
  - `MASK.BAD(F32)` specifies the threshold for bad pixels (in standard deviations); if negative, assume it's something like a Poisson distribution.

Mean statistics specified by a string (for `COMBINE`, `MEAN`) may be one of `MEAN`, `MEDIAN`, `ROBUST`, `FITTED` or `CLIPPED`. The standard deviation statistic (`STDEV`) may be one of `STDEV`, `ROBUST_STDEV`, `FITTED_STDEV`, or `CLIPPED_STDEV`.

#### 5.2.2.1 Example

```
# Recipe configuration for ppMerge

ROWS            S32     512 # Number of rows to read at once
ELECTRONS       F32     100.0           # Minimum number of electrons for useful signal
SAMPLE          S32     100             # Sampling factor for measuring the background
REJ F32 3.0 # Rejection threshold (sigma)
ITER S32 1 # Number of rejection iterations
FRACHIGH F32 0.0 # Fraction of high pixels to reject immediately
FRACLOW F32 0.0 # Fraction of low pixels to reject immediately
NKEEP S32 5 # Minimum number of pixels in stack to keep
FRINGE.NUM S32 10000 # Number of fringe regions
FRINGE.SIZE S32 5 # Half-size of fringe regions
FRINGE.XSMOOTH S32 5 # Number of smoothing regions in x
FRINGE.YSMOOTH S32 11 # Number of smoothing regions in y
SHUTTER.SIZE S32 128 # Size for shutter measurement regions
SHUTTER.ITER S32 1 # Number of iterations for shutter measurement
SHUTTER.REJECT F32 2 # Rejection limit for shutter measurement
MASK.SUSPECT F32 5.0 # Threshold for suspect pixels (sigma)
MASK.BAD F32 -4.0 # Threshold for bad pixels (sigma)
MASKVAL S32 0xff # Mask value for input data
COMBINE STR CLIPPED # Statistic to use for combination
MEAN STR ROBUST_MEDIAN # Statistic to use to measure the mean
STDEV STR ROBUST_STDEV # Statistic to use to measure the stdev
WEIGHTS BOOL FALSE # Use image weights?
```

### 5.2.3 PPSTATS

The `PPSTATS` recipe contains options for `ppStats` or its library used within another program:

- `SAMPLE(F32)` specifies the fraction of the cell to sample (for statistical measurements).
- `MASKVAL(U8)` specifies a mask value to use for the statistics.
- `HEADER(STR)` specifies headers (may be listed, separated by whitespace) to print. Multiple `HEADER` entries may exist, if it is declared `MULTI`.
- `CONCEPT(STR)` specifies concepts (may be listed, separated by whitespace) to print. Multiple `CONCEPT` entries may exist, if it is declared `MULTI`.
- `STAT(STR)` specifies statistics (may be listed, separated by whitespace) to print. Multiple `STAT` entries may exist, if it is declared `MULTI`. Acceptable statistics names are those parsed by `psStatsOptionFromString`.

#### 5.2.3.1 Example

```
### ppStats recipe for Phase 0 with MegaCam

# Options governing statistics
SAMPLE          F32     0.1     # Fraction of cell to sample
MASKVAL         U8      0xff    # Mask value to use for statistics

# Define the outputs as MULTI
HEADER          MULTI
CONCEPT         MULTI
STAT            MULTI

# Values to return
HEADER          STR     OBSERVER        # Observer name
CONCEPT         STR     FPA.OBJECT      # Object name
CONCEPT         STR     FPA.OBSTYPE     # Observation type
CONCEPT         STR     FPA.FILTER      # Filter
CONCEPT         STR     FPA.RA FPA.DEC  # Telescope pointing
CONCEPT         STR     FPA.AIRMASS     # Airmass
CONCEPT         STR     FPA.ALT FPA.AZ  # Telescopy alt/az
CONCEPT         STR     FPA.POSANGLE    # Rotator angle
CONCEPT         STR     CHIP.TEMP       # Detector temperature
CONCEPT         STR     CELL.EXPOSURE   # Exposure time
CONCEPT         STR     CELL.TIME       # Time of exposure
STAT            STR     ROBUST_MEDIAN   # Background estimator
STAT            STR     ROBUST_STDEV    # Background standard deviation estimator
```

### 5.2.4 PSPHOT

**EAM to fill this in. (TBD)**

### 5.2.5 PSASTRO

**EAM to fill this in. (TBD)**

# 6 Revision Change Log