

Mosaic Pipeline Operator's Manual

F. Valdes, N. Zarate

**National Optical Astronomy Observatory
Data Products Program**

Draft: November 2, 2006

Table of Contents

1	Introduction	1
2	The Role of the Operator	1
3	Basic Operation	2
4	Pipeline Scheduling Queue (PSQ)	2
4.1	Maintaining the PSQ	3
4.2	Interacting with the PSQ During Operations	3
5	Pipeline Monitor: pipemon	3
6	The Pipeline Blackboard	4
7	Reviewing Results	5
8	Generating Reports	5
9	Glossary of Terms	6

List of Figures

1	The pipeline status: plstatus	2
2	Running the pipeline: plrun	7
3	Stopping the pipeline: plstop	8
4	Pipeline scheduling queue: PSQ table	8
5	Pipeline scheduling queue interface: K4M table	9
6	Running the pipeline monitor: pipemon	9
7	Starting and stopping the switchboard server: serversb, stopserver	9
8	Example of the pipeline monitor showing processing from the Mosaic Pipeline	9
9	Example data products review page	10
10	Making pipeline reports: mkreportdb, mktxtreport	11

1 Introduction

This Mosaic Pipeline Operator's Reference Manual provides the basics for operating the Mosaic Pipeline. Details on the workings of the pipeline are found in other documents (PL001; PL002; PL003) and on-line manual pages. All the pipeline commands shown here have manual pages. Solutions to problems, particularly advanced problems, or common questions are found in the NOAO Help Desk at

<http://xarchive.tuc.noao.edu:8080>

In this manual we use the term *pipeline* in two ways. In some cases we will be referring to the Mosaic Pipeline *application*. While the operation of other pipeline applications, such as NEWFIRM, will be similar, this manual is specific to the Mosaic Pipeline and we will generally not include the word Mosaic. The second usage is to individual *pipelines* which are a set of modules performing some smaller piece of the processing. These pipelines are typically what is distributed among the processing nodes. There may be more than one instance of a type of pipeline.

2 The Role of the Operator

The operator of the pipeline is responsible for:

- maintaining the pipeline scheduling queue database
- starting the pipeline on the pipeline cluster
- stopping the pipeline
- monitoring the status of the pipeline system
- monitoring the status of the pipeline processing
- reviewing the data products
- reporting problems
- fixing problems with known solutions
- adding questions and resolutions to the help desk for future operators
- preparing reports

The ability to recognize problems and apply solutions is something that will increase with training and experience. In addition to posting questions and answers to the help desk database, operators can contribute additions and improvements to this manual.

3 Basic Operation

This sections describes the basic operation of the pipeline. It provides the simplest set of commands for operating the pipeline in the default mode. For some commands it does not go into detail about what is happening.

The first step is to log into the primary node, `pipedm`n of the pipeline cluster.

Check the available nodes and their status with the `plstatus` shown in figure 3.

Figure 1: The pipeline status: `plstatus`

```
pipedm$ plstatus

Pipeline node status report: Tue Oct 31 12:33:29 2006
Current node is pipedm

Node      Connected  NodeMgr  Available Pipelines
-----
pipen05   Yes        Down
-----
pipen04   Yes        Down
-----
pipen07   Yes        Down
-----
pipen06   Yes        Down
-----
pipen01   Yes        Down
-----
pipedm    Yes        Down
-----
pipen03   Yes        Down
-----
pipen02   Yes        Down
-----
pipen08   Yes        Down
-----
```

In this example all the pipeline processing nodes are up and no pipeline processes are running. Part of the commands that start and stop the pipeline include printing the status. So in the next step of running the pipeline note the output when everything is running.

To start the pipeline use the the `plrun` command shown in figure 3. The message about stopping the pipeline is ok in this context but if you know the pipeline is not running and wish to skip the step of stopping the pipeline add a `-s` flag.

To stop the pipeline the `plstop` command is used as shown in figure 3.

4 Pipeline Scheduling Queue (PSQ)

The pipeline scheduling queue is what drives the submission of data to the pipeline. The queue information is maintained in a (mysql) database. In order to fulfill the responsibility of maintaining the PSQ database you must understand the schema. This is described in (PL006).

4.1 Maintaining the PSQ

[TBD]

4.2 Interacting with the PSQ During Operations

The operator interface to this database is through a browser at the following address. Note that the line break is for formatting.

```
http://dpopnsn.tuc.noao.edu:8080/DPP/Operations/DCI/pipeline/  
PipelineProduction/PipelineMonitoringUpdate/summary_report_form1
```

The two types of tables which are accessed through the browser interface are a table of queues (see figure ??) and a table of datasets for each queue (see figure 4.2). The main operator field in the table of queues is the **State**. This has two values, enabled or disabled. By changed this the operator can turn a queue on or off. Note that any effect on the pipeline when it is running occurs the next time the pipeline checks for new data to process.

For a particular queue there is a table of datasets. The operator control field is the **Status**. It can have the values:

hold: Ignored by the pipeline.

pending: Can be selected by the pipeline to be processed.

submitted: Has been submitted to the pipeline.

completed: Completed by the pipeline.

The most common interactions for the operator are to change completed back to pending to force reprocessing. If the pipeline is stopped and restarted while a dataset is processing, say to recover from a crash, submitted datasets may be reset to pending. In this latter case however, the `plrun` command has a “-r” flag to reset submitted datasets back to pending. This is the recommended way to recover from a failure during processing.

5 Pipeline Monitor: `pipemon`

To monitor the pipeline processing, the tool `pipemon` is used. This requires that you are running an X server on the monitor and X forwarding is enabled. To start the pipeline monitor use this command. Figure 5 shows an example of starting the monitor.

Note that a process called the *switchboard server* is started automatically as needed. This server can remain running continuously. If you want to control the starting or stopping of the server use the following commands.

The pipeline monitor has two components. One is an `xgterm` that is created iconified. Depending on your X window manager it may be invisible or have an icon. The actual monitor is a child window of the this `xgterm`. The **Quit** button will not only exit the monitor window but also the `xgterm` window.

Figure 5 shows an example of the pipeline monitor.

The monitor updates whenever the pipeline *blackboard* changes. But this means if you start the monitor after the pipeline has been running you will not have all the information in the monitor. The **Read** button is used to load entire blackboard into the monitor. This may also be useful if you have deleted some information that you want to recover again.

There are three text entry boxes at the top of the monitor. These are used to manage the datasets displayed which can become quite larger. These boxes all take a regular expressions which is match against the entire blackboard. The expression is applied when a carriage return is entered while the cursor is in the box. Note that typing a carriage return when there is nothing in the box is equivalent to no pattern.

The **Show** entry selects lines to be displayed. Only lines that match this pattern will be shown. The **Hide** entry selects lines that are not to be shown. The hide expression has precedence over the show expression. When hiding lines the information remains in the monitor and can be seen again by changing the show and hide expressions. The **Delete** expression is used to remove any matching lines from the information in the monitor. Note that an empty pattern in this box will delete all lines so it is equivalent to a clear of the blackboard in the monitor. Any deleted information can be recovered (provided the pipeline is not stopped) by using the **Read** button.

One special form of regular expression is a parenthesized list of patterns separated by `|`. The initial setting of the **Hide** box makes use of this to hide datasets which are done or of less interest. It is common to hide completed dataset. By convention, the pipeline sets the last blackboard flag to “`d`” upon completion so the pattern “`d_`” is good for matching these datasets. In figure 5 the default hide expression was changed to show the completed datasets.

6 The Pipeline Blackboard

The status of *datasets* in the pipeline is shown by entries in the pipeline *blackboard*. The first part of the entry is the dataset name. The names are structured following a particular convention. The first part is *queue* name, the next is a dataset name in the queue, the third is a unique timestamp. After than are hyphen separated names where the first three letters are the pipeline and the rest is some identifier.

In figure 5 the second line has a dataset that has the following meaning. It is from a queue called CTEST, a dataset 3Z4F3O, and submitted to the pipeline with time stamp 1162318044. The ftr pipeline was called to process the I band exposures, the ngt pipeline was called to do the basic instrumental calibration, and the specific exposure is names ct4m200041017T002808. There is a further hidden sif pipeline for the individual CCDs of the exposure. In future this naming convention can be used to implement a more tree structure display.

The second field of the blackboard is the pipeline, the third is the node on which the pipeline is running, and the last field is the status of each module in the pipeline.

The status is represented by a single character so that each character in the string is for a module. Initially the status of a module is the underscore character.

To determine the name of each stage used the `stages` command as in the following example.

The first column is the stage number corresponding to the character in the blackboard status field and the second column is the stage name. The stage name is useful for finding the process log. In the third line of figure 5 notice that the sixth and eighth characters are “e”. This character is the usual flag for an error. By consulting the `stages` command we find that the errors occurred in the `mefgcat` and `mefwcs` stages. The `mefgcat` stage gets a coordinate catalog covering the exposure. For some reason it failed and it then follows that the WCS solution could not be performed by the `mefwcs` stage.

```
pipedm$ stages mef
```

```
Stages for mef pipeline:
```

```
1 mefstart
2 mefsetup
3 mefxtalk
4 meffrxtm
5 meffrsif
6 mefgcat
7 mefwace
8 mefwcs
9 mefacedq
10 meftrigsif
11 mefhdr
12 mefgif
13 mef2dsp
14 mefdone
```

7 Reviewing Results

When the processing of biases, processing of dome flats in each filter, and processing of science exposures in each filter are completed a review web page is constructed. These pages are available at the following URL.

```
http://pipedm/data2/MarioData/DPSDATA/output/html
```

Selecting a dataset will load a page with summary information and small preview graphics for the data products of the pipeline. An example is shown in figure 7. It is the role of the operator to periodically examine these review pages and, from experience note anything that appears wrong or unusual. In that case notify the designated supervisor or scientist along with the name of the review page (the group dataset) and the area of concern.

8 Generating Reports

While there may be additional reports that must be generated, one report is generated with a pipeline command to query the pipeline processing database. The command `mkreportdb`, shown in figure 8, accesses the pipeline database, extracts history information, and builds tables in an operations database. These tables may be used to build reports in other more refined formats.

However, `mkreportdb` will produce a simple text report. To reproduce the report without having to extract the history information again, a slow process, the command `mktxtreport` may be used. How often the report database and reports need to be generated is an operations and management requirement. Note that this may be automated and cron scheduled if desired.

9 Glossary of Terms

blackboard The collection of lines describing each pipeline dataset being processed or completed.

The blackboard includes the dataset identifier, the pipeline name, the node on which it is running, and the set of processing flags, one per stage.

dataset A logical identifier for the set of data processed by a pipeline. A logical dataset may be related to any number of actual Mosaic exposures or pieces of exposures. The meaning of a dataset differs for each pipeline. One may think of the datasets as forming a hierarchy starting with data from an entire block of nights down to individual amplifiers and CCDs.

pipeline A single set of modules performing some processing or data handling function. A set of pipelines make up a pipeline application such as the Mosaic Pipeline.

pipeline application The complete set of pipelines which do a particular type of processing. The Mosaic Pipeline is a pipeline application.

stage A single step in a pipeline associated with a particular action. Each stage has an associated status flag in the blackboard.

References

[PL001] Valdes, et al., 2006, The NOAO High Performance Pipeline System, NOAO DPP Document PL001

[PL002] Valdes, et al., 2006, The NOAO Mosaic Camera Pipeline System, NOAO DPP Document PL002

[PL003] Valdes, et al., 2006, The NOAO Mosaic Camera Pipeline: Methods, Data Products, and Science Verification System, NOAO DPP Document PL003

[PL006] Valdes, F., 2006, The Pipeline Scheduling Queue Database, NOAO DPP Document PL006

Figure 2: Running the pipeline: plrun

```

pipedmn$ plrun
=====
Starting the pipeline
Tue Oct 31 12:40:06 MST 2006
Data manager node = pipedmn
Processing nodes = pipen01,pipen02,pipen03,pipen04,pipen05,
  pipen06,pipen07,pipen08
Stopping any current pipeline processes...
Cleaning pipeline processing data directories...
Starting the name, directory, and data manager servers...
Starting the node managers...
Setting pipeselect strategy...
Starting the pipelines...
Queuing next pending dataset ...

Pipeline node status report: Tue Oct 31 12:40:54 2006
Current node is pipedmn

Node      Connected  NodeMgr  Available Pipelines
-----
pipen05   Yes        Running  cal, day, frg, mdp, mef, pgr,
rsp, scl, sft, sif, xtc, xtm
-----
pipen04   Yes        Running  cal, day, frg, mdp, mef, pgr,
rsp, scl, sft, sif, xtc, xtm
-----
pipen07   Yes        Running  cal, day, frg, mdp, mef, pgr,
rsp, scl, sft, sif, xtc, xtm
-----
pipen06   Yes        Running  cal, day, frg, mdp, mef, pgr,
rsp, scl, sft, sif, xtc, xtm
-----
pipen01   Yes        Running  cal, day, frg, mdp, mef, pgr,
rsp, scl, sft, sif, xtc, xtm
-----
pipedmn   Yes        Running  dir, dps, dts, ftr, ngf, psq
-----
pipen03   Yes        Running  cal, day, frg, mdp, mef, pgr,
rsp, scl, sft, sif, xtc, xtm
-----
pipen02   Yes        Running  cal, day, frg, mdp, mef, pgr,
rsp, scl, sft, sif, xtc, xtm
-----
pipen08   Yes        Running  cal, day, frg, mdp, mef, pgr,
rsp, scl, sft, sif, xtc, xtm
-----
Pipeline started successfully
=====

```

Figure 3: Stopping the pipeline: plstop

```

pipedmn$ plstop
=====
Stopping the pipeline
Tue Oct 31 12:46:11 MST 2006
Data manager node = pipedmn
Processing nodes = pipen01,pipen02,pipen03,pipen04,pipen05,
  pipen06,pipen07,pipen08
Stopping any current pipeline processes...

Pipeline node status report: Tue Oct 31 12:46:15 2006
Current node is pipedmn

Node      Connected  NodeMgr  Available Pipelines
-----
pipen05   Yes        Down
-----
pipen04   Yes        Down
-----
pipen07   Yes        Down
-----
pipen06   Yes        Down
-----
pipen01   Yes        Down
-----
pipedmn   Yes        Down
-----
pipen03   Yes        Down
-----
pipen02   Yes        Down
-----
pipen08   Yes        Down
-----
Pipeline stopped successfully
=====

```

Figure 4: Pipeline scheduling queue: PSQ table

Update Pipeline PSQ Report

Total Number Of Records Found : 2

PSQ Name	Queue	Data	Pipeline	State	Query
C4M	C4M	C4MDATA	dir	enabled	instrument in ('mosaic_2','optic') and imagedt='focus' and ofile not like '%test%'
K4M	K4M	K4MDATA	dir	enabled	instrument='mosaic_1' and imagedt='focus' and ofile not like '%test%'

[Home](#)

Figure 5: Pipeline scheduling queue interface: K4M table

Update Pipeline Monitoring Report For The Queue **K4M**

Total Number Of Records Found : 88

[1] 2 3 4 5

Dataset	Priority	Status	Submitted	Completed
20040116T	1	pending	None	None
20040117	1	pending	None	None
20040121	1	pending	None	None

Figure 6: Running the pipeline monitor: pipemon

```

pipedmn$ pipemon
Switchboard server is starting... [1] 28851
Starting the pipeline monitor... [1] 28853
    
```

Figure 7: Starting and stopping the switchboard server: serversb, stopserver

```

pipedmn$ serversb
Switchboard server is starting... [1] 28889
pipedmn$ stopserver serversb
Stopping serversb... done
    
```

Figure 8: Example of the pipeline monitor showing processing from the Mosaic Pipeline

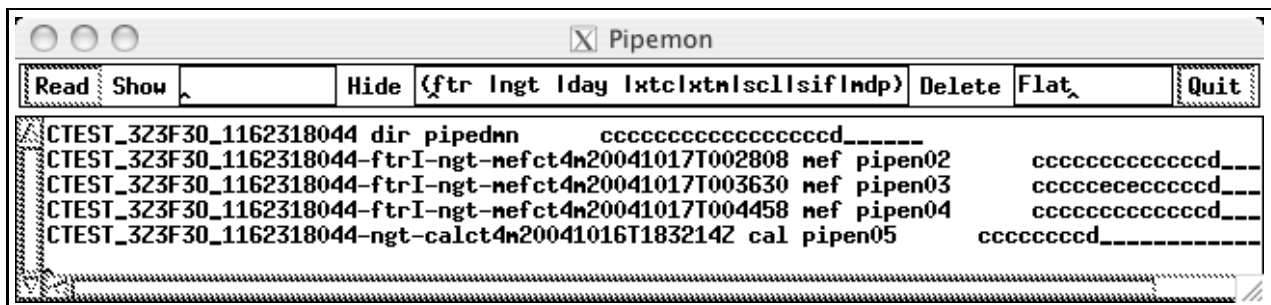


Figure 9: Example data products review page

CTEST_3Z3F3O_1162318044-I

[Sky Analysis](#)

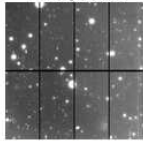
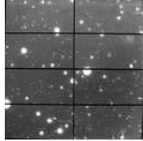
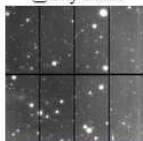
waa1 waa1.041017_0026.1038 I c6028 OEXPTIME = 400 A Next Generation Microlensing... MAXOBJAREA = 250996 SKYFRAC = 0.883 SKYMEAN = 35.6 AIRMASS = 1.2 USNO Match = 1707 MAGZERO(i) = 26.61 PHOTDPTH(i) = 22.84 SEEING = 1.0	<input type="checkbox"/> Sky Stack  ct4m20041017T002808 [800, x2]	 ct4m20041017T002808_r [800, x2]
waa2 waa2.041017_0035.1039 I c6028 OEXPTIME = 400 A Next Generation Microlensing... MAXOBJAREA = 342104 SKYFRAC = 0.899 SKYMEAN = 34.2 AIRMASS = 1.2 WCS/Phot Calibration Failed	<input type="checkbox"/> Sky Stack  ct4m20041017T003630 [800, x2]	

Figure 10: Making pipeline reports: mkreportdb, mktxtreport

```

pipedm$ mkreportdb
Copying PMAS database ...
Extracting report data ...
Creating report database ...

```

REPORT BY DATASET

Dataset	ninimages	insize	noutimages	outside	Time
UBVRI_C20041208_1160391078	331	46165	66	8181	NULL
UBVRI_C20041208_1160457520	126	17574	380	19999	6:33
UBVRI_C20041213_1160481121	194	27057	231	14613	6:42
UBVRI_C20041218_1160505321	299	41702	488	28762	12:22
UBVRI_C20050207_1160549926	297	41423	32	4210	NULL
UBVRI_K20041008_1160172527	124	15599	200	13688	3:31
UBVRI_K20041008_1160348100	66	7839	168	9449	2:23
UBVRI_K20041008_1160419314	66	7839	168	9449	2:34
UBVRI_K20041012_1160185219	165	19597	255	16884	NULL
UBVRI_K20041012_1160356713	118	13308	281	16904	3:28
UBVRI_K20041012_1160428562	117	13174	279	16639	3:59
UBVRI_K20041103_1160369210	254	33985	301	20509	6:3
UBVRI_K20041103_1160442909	95	12711	275	17038	4:3

REPORT BY MJD DAY AND QUEUE

MJDDay	Queue	ninimages	insize	noutimages	outside
54014	UBVRI	289	35196	200	13688
54015	NULL	NULL	NULL	255	16884
54016	UBVRI	184	21147	172	9985
54017	UBVRI	863	113874	1106	72989
54018	UBVRI	619	86333	1050	62515
54019	UBVRI	297	41423	341	20264

REPORT BY MJD WEEK AND QUEUE

MJDWeek	Queue	ninimages	insize	noutimages	outside
7716	UBVRI	289	35196	455	30572
7717	UBVRI	1963	262777	2669	165753

REPORT BY CALENDAR MONTH AND QUEUE

Month	Queue	ninimages	insize	noutimages	outside
2006-10	UBVRI	2252	297973	3124	196325

REPORT BY CALENDAR YEAR AND QUEUE

Year	Queue	ninimages	insize	noutimages	outside
2006	UBVRI	2252	297973	3124	196325