Scheduler Manual
D. Rabinowitz 2025 Oct 30

Contents

1. **Overview**

The LS4 scheduler reads in a plan of observations (aka “fields”) from a text file, and sends commands to the control programs for the telescope (questctl) and the LS4 camera (ls4_ccp) to implement the observations. The scheduler determines which fields will be observed and their order depending on codes in the observing plan and observational constraints specific to the type of observation.

Start and Stop

Typically, the scheduler is started by a control script (obs_control) that also starts up the control programs for the telescope and camera. When the scheduler is started, five command-line arguments must also be provided:

```
scheduler [obsplan] [yyyy] [mm] [dd] [verbose_flag]
```

where “obsplan” is the name of the obsplan file, “ yyyy mm dd” is the UT date for the following morning, and verbose_flag (0, 1,or 2) specifies the detail of message logging.

To stop the scheduler (leading to an exit) or to pause the scheduler (temporarily suspending observations), a Linux signal must be send to the scheduler (by sending a signal (SIGTERM stop, SIGUSR to puase) to the process. See “pause_scheduler” and “stop_scheduler” scripts.

Main Event Loop

At the highest level, scheduler implements the following main-even loop:

```
Read in and initialize observation plan.

Wait until sun goes down, dome opens, and telescope is ready to command.

While not yet morning twilight:

    if telescope ready:

      if stop requested:
         stop telescope tracking

      else if bad weather:
         stow the telescope and close dome

      else if dome is closed or scheduler is paused:
         wait 10 sec

      else:
```

```
            select next field to observe.
            Point telescope and take exposure.

        else:
          wait 10 sec

    Stow telescope, close dome and exit
```

2. **Observation Plan**

Each line of an observation plan ("obsplan") specifies the following:

ra, **dec** : telescope pointing (hours, deg)

expt : exposure time (sec)

shutter code : (explained below)

n_required : required number of exposures

interval : requested exposure interval (sec)

survey code (explained below).

comment (extra info for reference purposes)

Survey Codes (integer from 0 to 4):

0: no survey (generic observation)

1: Trans-Neptunian Object survey

2: Supernova survey

3: Must-do high-priority code

4: Ligo event followup

Shutter Codes (upper or lower case):

N: dark field (see note below)

Y: sky field

F: focus field

P: pointing-calibration field

E: evening twilight flat

M: morning twilight flat

L: dome flat

Note: darks with dec = -1, 0, 1 deg will be observed in evening twilight, dark time, and morning twilight, respectively. Otherwise, there is no time restriction.

3. **Observation Selection**

Before each observation, the scheduler updates the status of every planned observation. The relative status of the observations and their selection constraints determine the next observation to make.

Selection Constraints

- **pointing limits** (e.g. limiting airmass, hour angle, galactic latitude, moon separation)
- **exposure requirements** (expt, n_required, interval)
- **observable sky** (visible area of night sky).
- **interruptions** (dome closed for weather or maintenance).
- **observation cadence** (minimum time between observations).
- **survey code** (survey-specific constraints)

Field Descriptor

Internally, the scheduler maintaints a field descriptor for every planned observations. This records the current status and observing parameters (see "Field" structure in "scheduler.h"):

- **survey code** (from obsplan)
- **shutter code** (from obsplan)
- **status code** (see descriptions below)
- **pointing info** (**ra, dec, galactic lat, long, lunar separation**)
- **exposure info** (**expt, n_required, interval**)
- **rise/set times** (UT hours and Julian Days **ut_rise, ut_set, jd_rise, jd_set**)
- **expected time of next exposure** (Julian Date **jd_next**).
- **number of completed exposures** (**n_done**)
- **up time** before field sets or sun rises (**time_up**)
- **completion time** for remaining observations (**time_required**)
- **time left** before time_required exceeds time_up (**time_left**)
- **completed exposure info** (**filename, ut, jd, lst, ha, am, actual_expt**)
- **doable flag** (1/0 if the field will/won’t be observable in the remaining time)

status codes:

too_late : observable, but not enough time for remaining required exposures

not_doable: observations are not longer possible of this field

ready : the field can be observed now

do_now : the field can be observed and is of highest priority

Selection Algorithm

init_fields (start of night):

Initialize field descriptors for every observation.
Set n_observable to number of doable fields.

get_next_field (each time a new observation is made):

Loop through the field descriptors of every planned observation in two passes.

Pass 1:

update_field_status:

Set doable_flag=0 and status_code to “not_doable” for all fields satisfying any of the following:

-now completed (n_done = n_required)
-outside observing limits
-not ready to be re-exposed

Evaluate parameters to be used to select the next field:

Indices of earliest doable fields meeting specific criteria:

Field Index	Field Criterion
i_min_do_now	sky fields with do-now status
i_min_dark	dark fields with do-now status
i_min_flat	dome or sky flat

Tallies of doable fields meeting specific criteria:

Tally Name	Field Criterion
n_ready_must_do	ready status, must-do survey code
n_late_must_do	too-late status, must-do code
n_ready	ready status, not must-do code
n_do_now	do-now status, any survey code
n_late	too-late status

Minimum values for parameters of fields meeting specific criteria

Minimum Name	Field Parameter	Field Criterion
n_left_min_must_do	n_remaining	ready status, must-do code
n_left_min	n_remaining	ready status, not must-do code

Pass 2:

Consider each of following criteria (in order) until one is found with matching doable fields. Select the next field to observe from those matches. For criteria 3, 4, and 5, select the field that appears earliest in the obsplan. For criteria 1, 2 and 7, choose the field with the least time_left. For criteria 8, choose the field with the most time left assuming (and exceeding 0).

criterion 1: **status = “ready” and survey_code = “must-do”**

Criterion 2: **status = “too_late” and survey_code = “must-do”**[1]

Criterion 3: **status=”do_now” and shutter_code = dome, evening, or morning flat**

Criterion 4: **status=”do_now” and shutter_code = dark**

Criterion 5: **status=”do_now” and not (dark or flat)**

Criterion 6: **doable** and paired with previously observed field[1,2]

criterion 7: **status = “ready” and survey_code not “must-do”**

criterion 8: **status = “too_late” and survey_code not “must-do”**[1]

Notes:

1. the considered fields will have their exposure interval shortened so that completion_time = time_up (all remaining exposures become observable)

2. paired fields are consecutive fields in the obsplan that are both sky exposures (shutter code “Y”) and for which the separation in the sky parallel to the equator is less than a fixed interval (RA_STEP0 = 0.05 deg). This is meant to fill the gaps between CCDs in the LSQ camera.

4. **Observing Cadence**

The cadence of the observations depends on the staging of 4 key operations:

(1) exposure time (time shutter is open).

(2) image readout time (~20 sec for dual-amp readout, ~40 sec for single-amp readout).

(3) transfer time of the image data from the controllers to the host computer (~5 sec).

(4) the time to reposition the telescope and dome for the next observation (~ 20 sec).

Three cadences are possible, but only cadence 1 has been fully implemented.

cadence 1. linear staging

The simplest way to stage the operations is one at a time.

```
expose -> readout -> transfer -> slew -> ....
```

cadence 2. Parallel slew

A faster cadence is achieved by staging the telescope slew in parallel with the image readout and transfer :

```
expose -> readout -> transfer -> exposure -> readout -> transfer -> ...
          slew --------------               slew --------------
```

cadence 3. Parallel slew and transfer

The optimal cadence is achieved by parallel staging both slew and transfer with readout and expose. This is possible because the camera controllers have multiple buffers in memory allowing transfer of previously read image data while new data are being exposed and read out:

```
expose -> readout -> expose -> readout -> expose -> readout -> ...
          slew -----           slew ----            slew -----
transfer------------ transfer------------ transfer--------------
```

5. **Installation**

Server Setup

Create two users, “ls4” and “observer”. Make the default shell “tcsh” for the observer user. Only the “ls4” user will be able to edit the code and change the configuration parameters Only the “observer” user will be able to run the observing code.

Create a data directory, “/data”, with subdirectory “observer” with read/right/ownership for user “observer”.

Login as “ls4”. Create a directory (e.g. */home/*ls4/code) for code respositories. Install “ls4-scheduler”,”quest-src-lasilla”, and “ls4_control” from git hub. Follow the README instructions to compile. Make sure the read/write/execute permissions for the code directory allow execution and reading by anyone, but writing only by the owner (“ls4).

Edit file “observer_setup.csh” in ls4-scheduler/observer_conf. Change the setting for “LS4_HOME” to path for the code repository setup above (e.g. “/home/ls4/code”).

Log in as “observer”.
Make sure observer is running the “tcsh” shell.
Copy observer_setup.csh to /home/observer
link observer_dot_login to /home/observer/.login
Source home/observer/.login

Execute “observer_setup.csh”. This should create directory “~/bin” with links to all the code required for observing.

6. Configuration:

All configurable parameters are set by logging in as “ls4” and changing the value of environment variables set by “observer_dot_login”. Hard coded parameters must be changed by editing “scheduler.h” and/or “scheduler.c” and then recompiling scheduler. Most environment variables should not be changed, except for the following purposes:

Single or Dual amp camera readout:

```
# select “left”, “right”, or “both”.
observer_dot_login: setenv CCD_AMP_SELECTION “both”
```

Verbosity Level:

```
observer_dot_login:
  # choose 0, 1, or 2
  setenv VERBOSE_FLAG  1

  # choose “INFO”, “DEBUG”, “WARN”, or “ERROR”.
  # INFO is minimal verbosity, DEBUG is more verbose,
  # ERROR is error messages only, WARN is warning and error messages only.
  setenv CCP_LOG_LEVEL “INFO”
```

Daytime testing:

```
observer_dot_login: setenv OVERRIDE_DOME  1
scheduler.h       : #define USE_ANYTIME_START 1
```

Simulated Operations :

```
observer_dot_login:
  FAKE_OBS       :  1 to simulate telescope and/or camera operations, 0 otherwise
  FAKE_TELESCOPE :  1 to simulate telescope, 0 otherwise
  FAKE_CAMERA    :  1 to simulate camera
  FAKE_UT_OFFSET :  offset to UT in simulation mode
```

Observation Cadence:

Choose False for faster cadence. This allows slewing telescope while reading out and transferring image from controllers to host. However, this option cause errors when last tested.

```
scheduler.c    : #define WAIT_FLAG True
```

Twilight Flat and Dark scheduling:

```
scheduler.h:

  # time after/before sunset/sunrise (hours) before twilight flats can start
  #define SKYFLAT_WAIT_TIME (0.5/24)

  # time after/before sunset/sunrise (hours) when darks must stop
  #define DARK_WAIT_TIME (0.0/24) /* wait half hour after sunset or stop
```

Observing Limits:

Various observing limits are hard-coded in scheduler.h

```
#define MIN_MOON_SEPARATION 15.0 /* minimum target-moon separation (deg) */
#define MAX_AIRMASS 2.0
#define MAX_HOURANGLE 4.3
#define MAX_OBS_PER_FIELD 100 /* maximum number of observations per field */
#define MAX_FIELDS 500  /* maximum number of fields per script */
#define OBSERVATORY_SITE "La Silla"
#define MAX_EXPT 1000.0  /* sec */
#define MAX_INTERVAL (43200.0/3600.0) /* hours */
#define MIN_INTERVAL /*(900.0/3600.0)*/ 0.0 /* hours */
#define MIN_DEC -89.0 /* no decs lower than this (deg) */
#define MAX_DEC 30.0 /* no decs higher than this (deg) */
#define MIN_FOCUS 24.0 /* no focus setting (mm) less than this */
#define MAX_FOCUS 30.0/* no focus setting (mm) more than this */
#define MIN_FOCUS_INCREMENT 0.025 /* no focus increment (mm) less than this */
#define MAX_FOCUS_INCREMENT 0.10 /* no focus increment (mm) less than this */
#define MAX_FOCUS_CHANGE 0.3 /* maximum change from expected default mm */
```

```
#define MAX_BAD_READOUTS 3 /* quit trying exposure after this many failures */
#define CLEAR_INTERVAL 0.1 /* hours since last exposure to start  clear */
#define USE_12DEG_START 1 /* 1 or 0 to start at 12- or 18deg twilight */
#define USE_ANYTIME_START 1 /* Start anytime, ignore sun altitude */
```

7. **Operation**

Remote and Local operations:

1. Log in to ESO VPN.
2. Connect to the vnc server on ls4-nuc (134.171.80.150:5902)
3. Open a terminal window
4. Log in to ls4-workstn as observer (use “observer” alias)
5. Start the telescope controller using “start_questctl”

(note: controller will report errors until telescope control is given to questctl using “alt 1” on TCS keyboard).

6. Log out from ls4-workstn.
7. Enter “observer”.

This will launch several windows showing the message logs from questctl, ccp (the camera controller), the scheduler, and an obs-control script

8. Use obs-control window start/stop scheduler and ls4_control (aka “ccp”)
9. Use “opendome_raw” and “closedome” to open/close dome. Closing the dome still pause the scheduler until the dome is opened again.

Engineering:

Login to ls4-workstn as observer.
Use “start_questctl” and “stop_questctl” to start/stop telescope controller.
Use “obs-control” to start/stop scheduler.
Use “restart_ccp” to start/stop camera controller.