Towards Automating Operations of SGP VGOS Stations

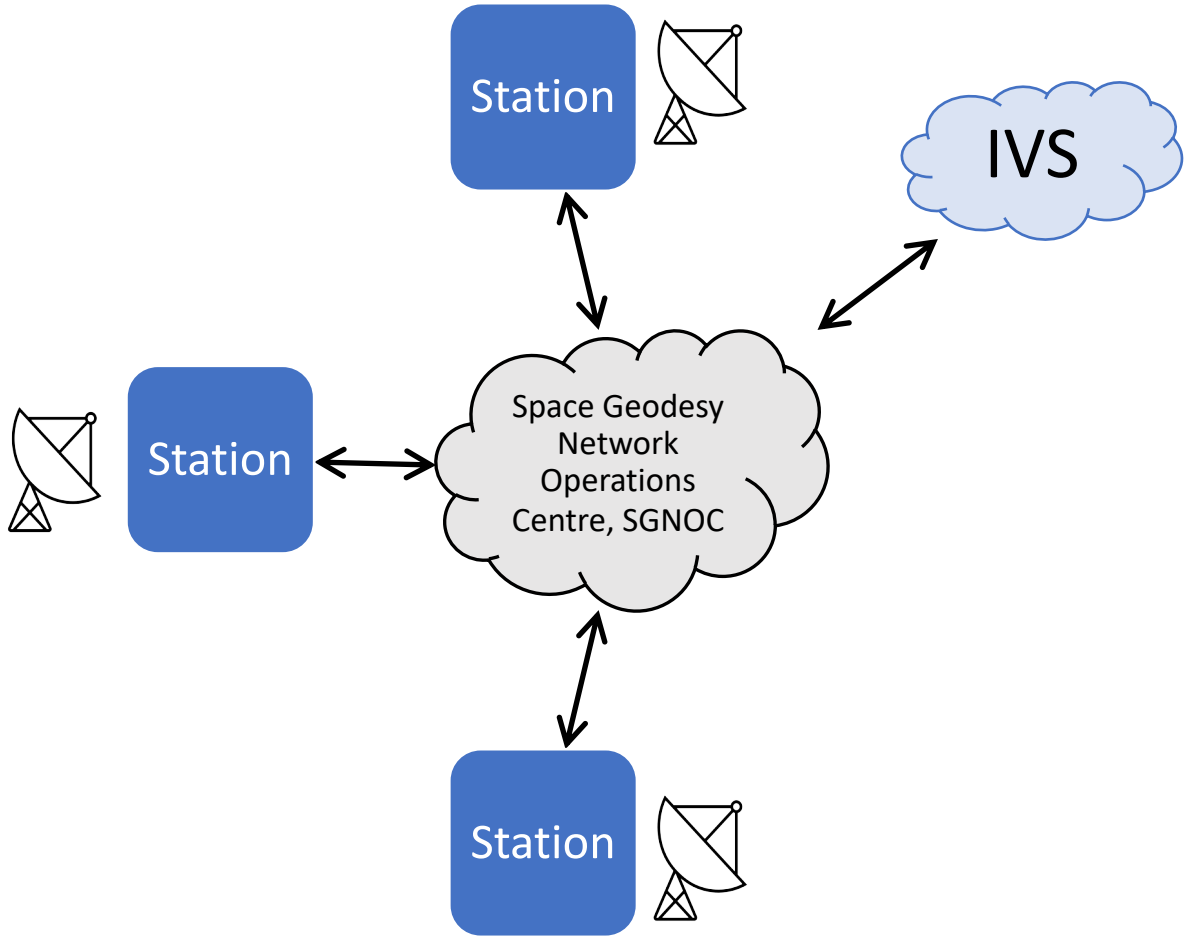
Jim Lovell, NVI (lutruwita/Tasmania outpost)
jejlovell@gmail.com

Darryl Lakins, NASA GSFC

12th IVS General Meeting, 28 March – 1 April 2022

Introduction: The Next Generation Field System (NGFS)

- Drivers:
  - VGOS: Broadband feeds and receivers, digital back ends and high bandwidth networks
  - Up to 10 stations in the NASA Space Geodesy Project (SGP) network, controlled and monitored centrally with stations unattended



Introduction: The Next Generation Field System (NGFS)

- Aims:
  - Support SGP VGOS stations,
  - Can be effectively maintained,
  - Adaptable to future technological developments (extensible),
  - Capable of a high level of automation.

NGFS Development Path

While the NGFS is being built, we are developing the techniques required for automation, using the existing Field System and a test VLBI Operating Centre (VOC).

Two initial applications at the station end of the operations network:

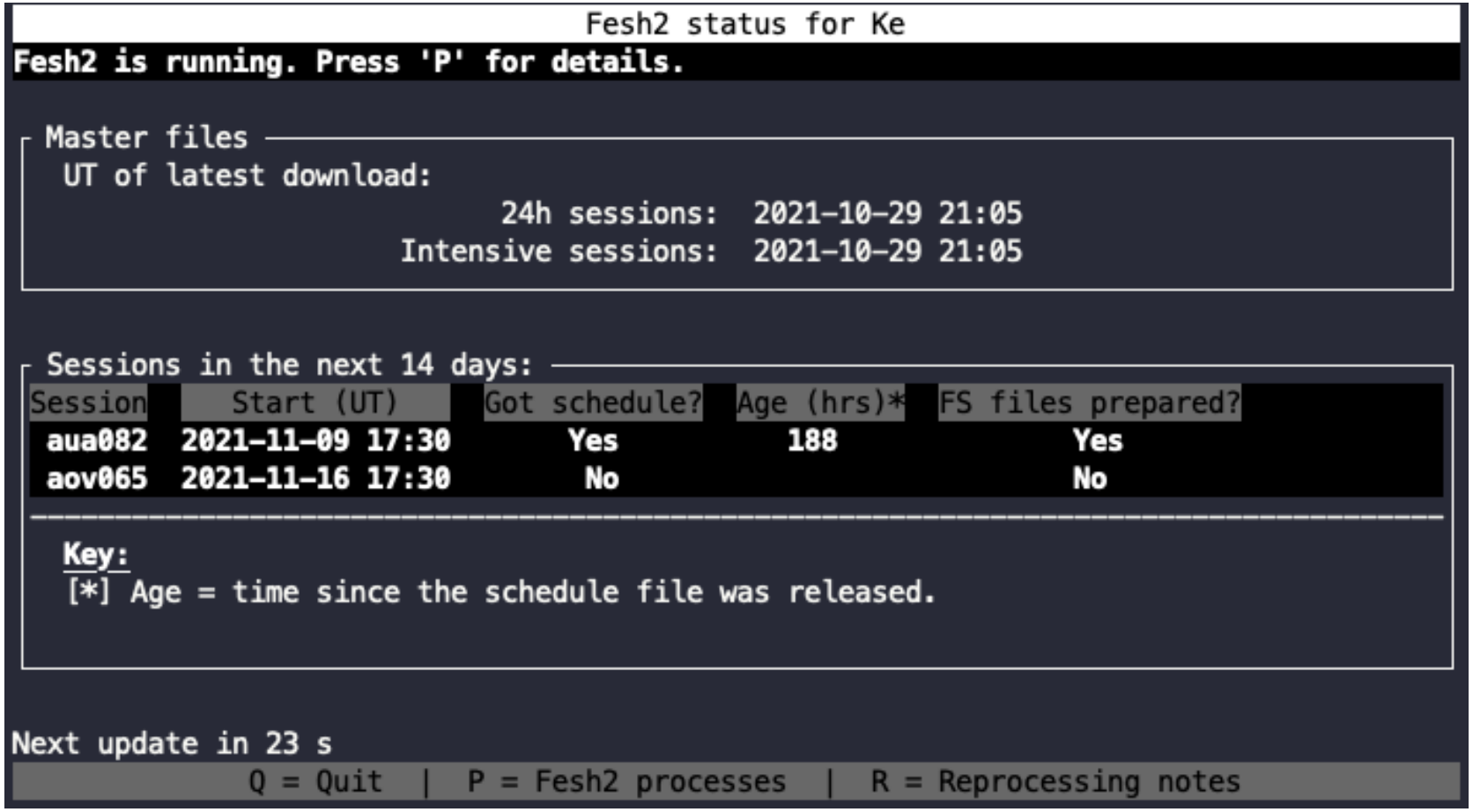
1. Fesh2
2. SGPAutomate

Testing on Field System at Hobart (thanks!), then SGP sites

Fesh2: Automated schedule file preparation at the station

- Keeps a local, current version of Master files
- Checks all IVS servers for latest schedule file
- Automatic processing with Drudg
- Can be run once or in monitoring mode with continually updating status reports
- Checks if local files have been modified and won’t overwrite by default.
- Can run as a foreground task or in background as a service
- Compatible with Python 3.5 and above

Fesh2 in action



Where do I get Fesh2?

- May be released with the next FS update (10.1)
- Or if you can’t wait, get in touch: jejlovell@gmail.com

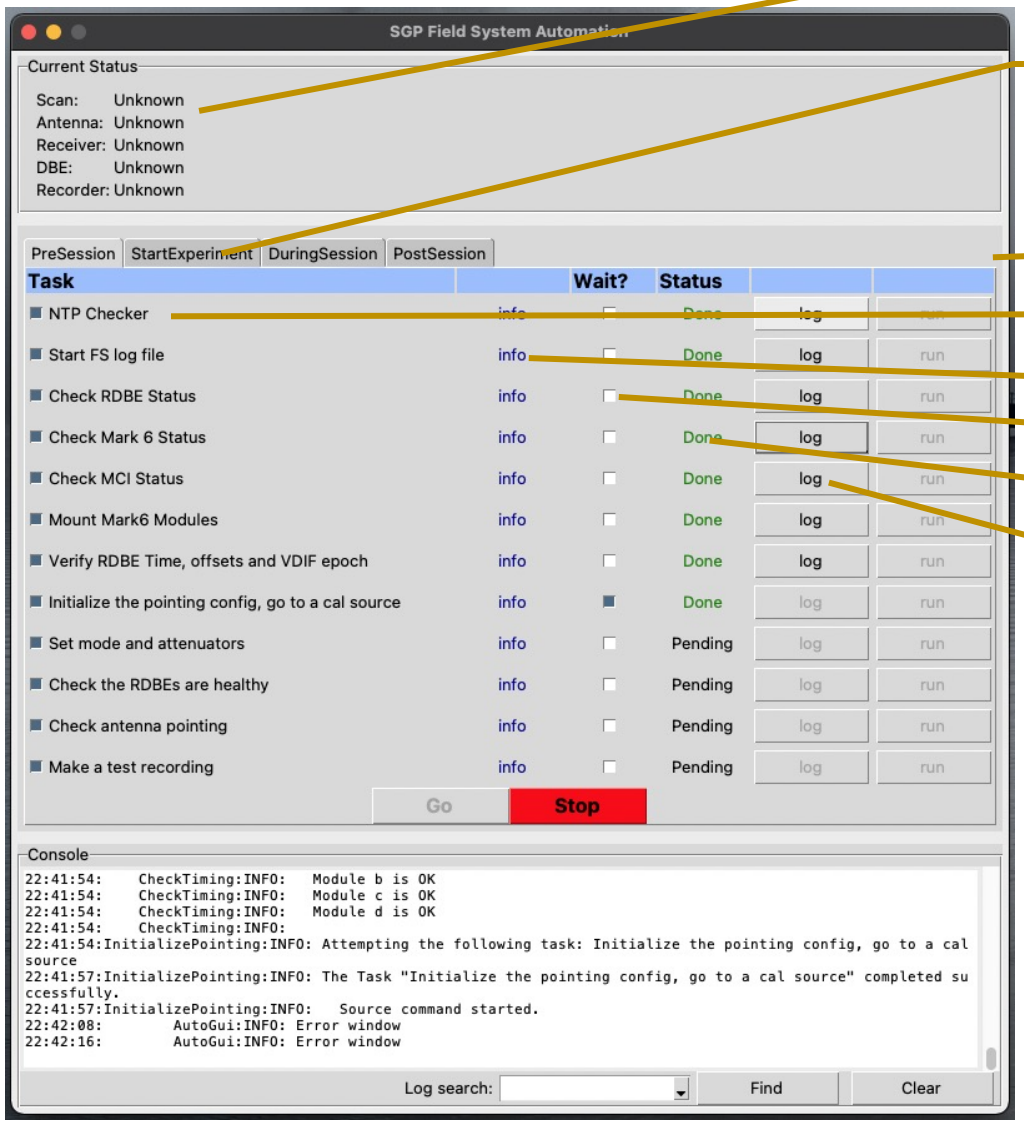
Next steps

- A VOC version that will:
  - Notify SGNOC of schedule status at the station
  - Receive push notifications of schedule changes rather than regular polling of IVS servers
  - Proof-of-concept only at this stage

2. SGPAutomation

- Software is intended to be highly configurable to suit individual station needs.
- It covers:
  - Pre-session procedures and checks
  - Start session procedures
  - In-session checks and monitoring
  - Post-session procedures

SGPAutomation GUI main page



Status summary

One tab per stage

Task list

Task to do (selectable)

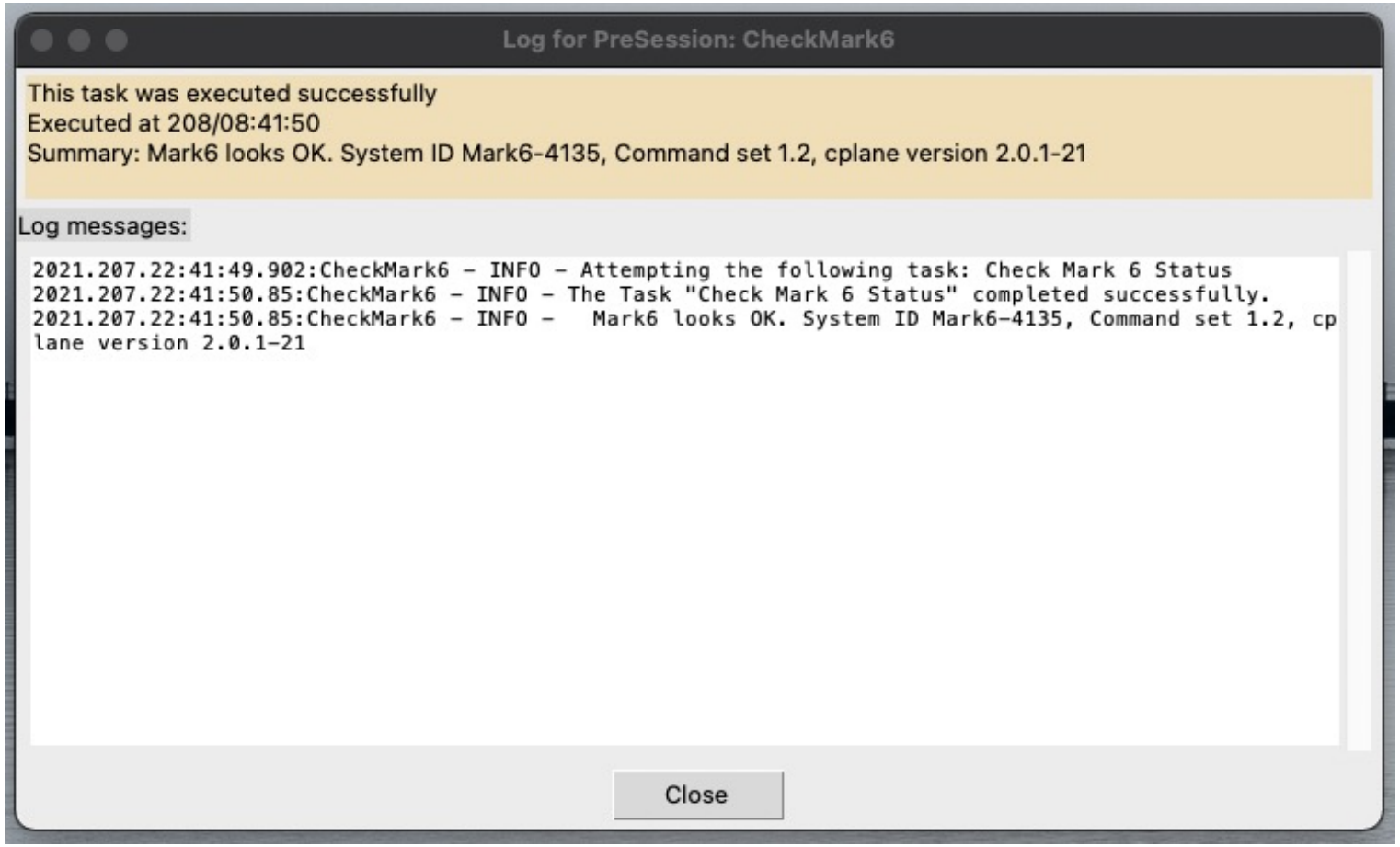
Documentation

Wait before continuing?

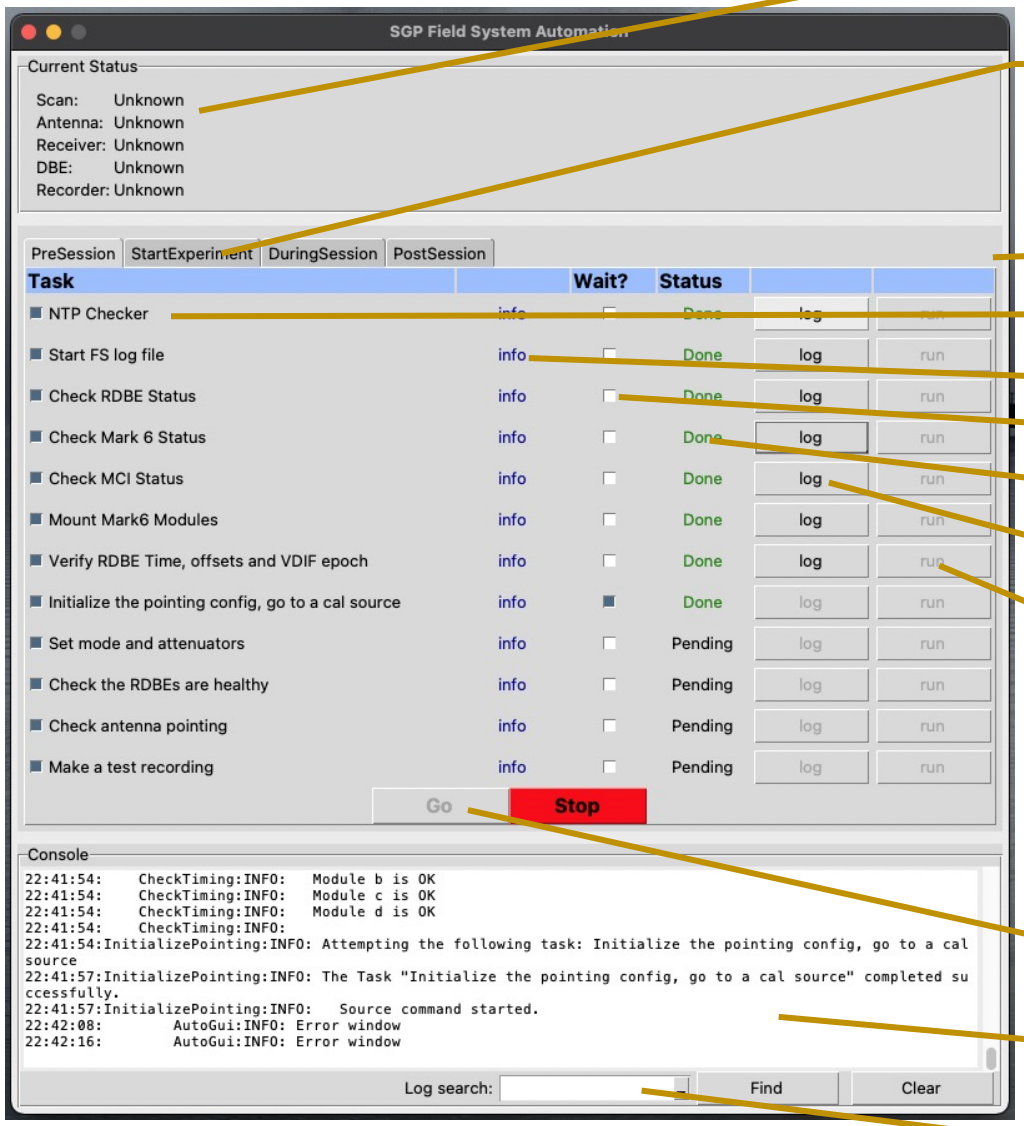
Status

View log

Summaries of individual task activity can be shown via a the ‘log’ button.



SGPAutomation GUI main page



Status summary

One tab per stage

Task list

Task to do (selectable)

Documentation

Wait before continuing?

Status

View log

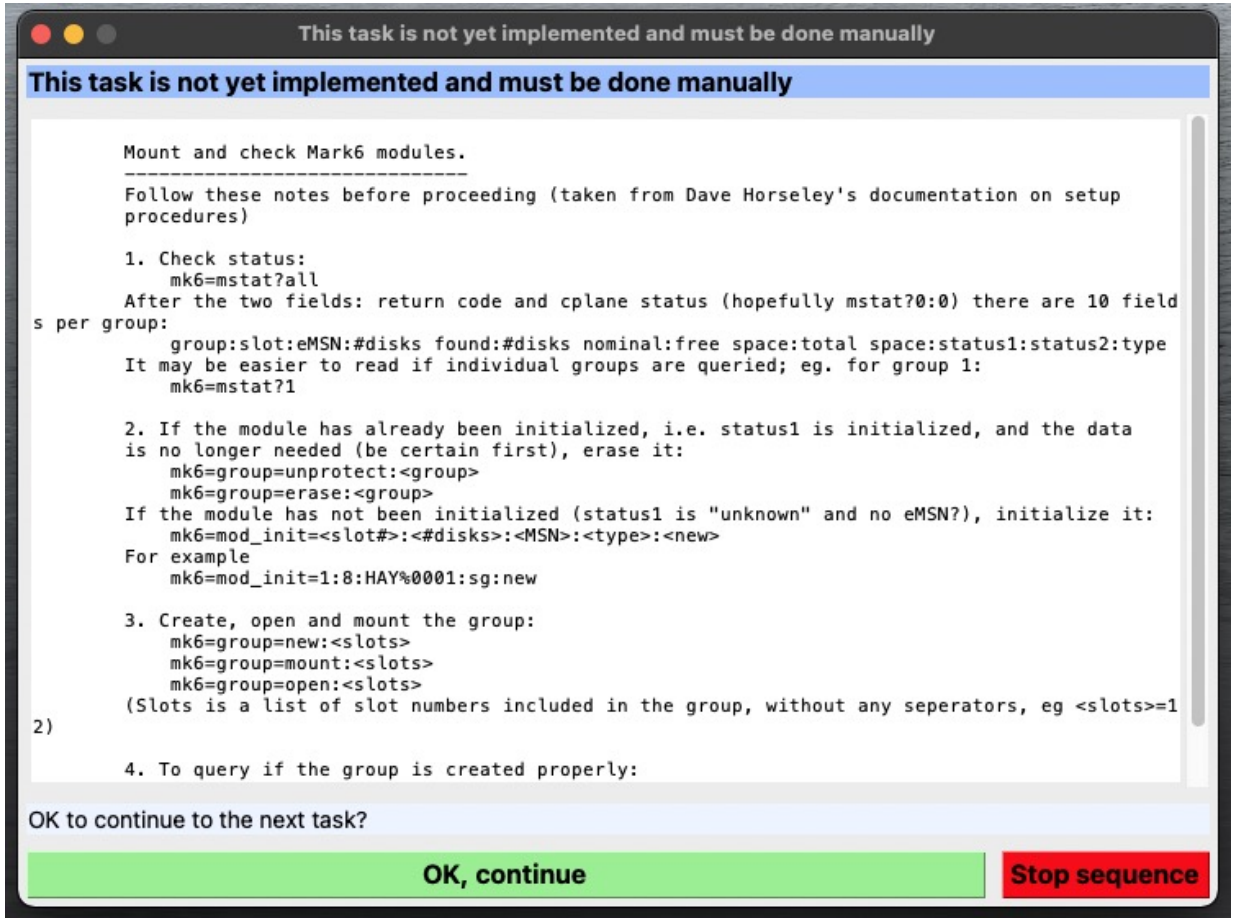
Run the task

Start/stop task sequence

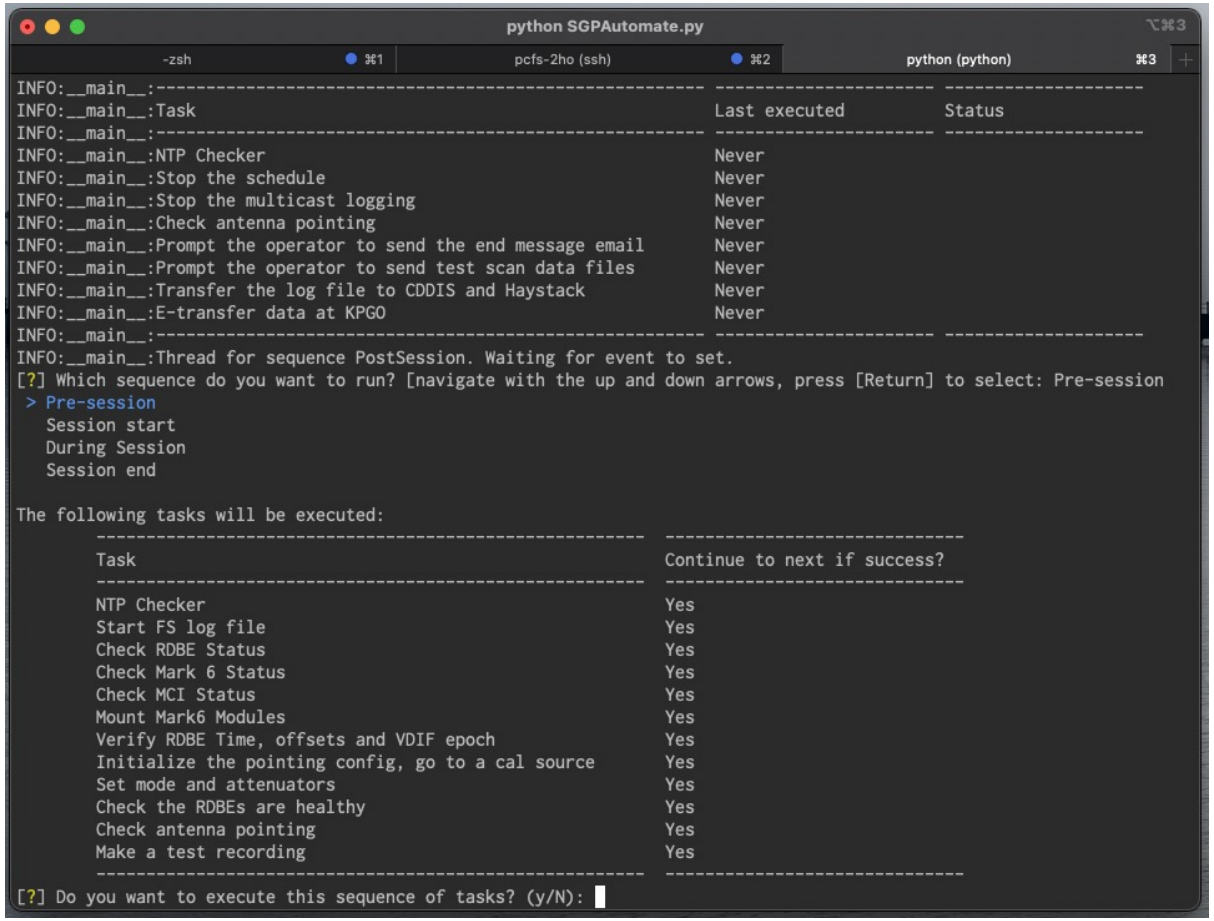
Log

Search log messages

When operator input is required, a dialog is shown.



The text-based interface is more limited than the GUI but is available if preferred.



SGPAutomate is configurable

You can configure the tasks to be done for each stage (e.g. a task list for pre-session)

Commented template code is provided to help in writing new tasks

What next?

- VOC communications
- Complete implementation of main tasks for SGP sites
- Eventually make available through FS distribution.

Thankyou.

"… standing on the shoulders of Giants."
– I. Newton

Configure tasks and task sequences

Name task, description and config

Order tasks in a sequence. Can be re-used

```
[tasks.StartFSLogFile]
    name = "Start FS log file"
    description = '''
    Open a FS experiment log. The current or next session, determined from the Master file, is used by
    default but the user can enter a name manually
    '''
    timeout = 3.0
    continue_if_ok = true

[tasks.CheckRDBE]
    name = "Check RDBE Status"
    description = '''
    Runs the command rdbe_statu
    '''
    timeout = 3.0
    continue_if_ok = true

[tasks.CheckMark6]
    name = "Check Mark 6 Status
    description = '''
    Issue the command mk6=dts_i
    that the cplane and dplane
    '''
    timeout = 3.0
    continue_if_ok = true
```

```
[sequences]
    # Arrays contain a list of tasks (given above) in the order they should be executed for a a
    # specific activity For example, the PreSession sequence contains all tasks to be carried out
    # before a sesion starts
    [sequences.PreSession]
        name = "Pre-session"
        description = "Procedures to be carried out before the session."
        repeating = false
        repeat_gap_min = 0
        tasks = ["CheckNTP", "StartFSLogFile", "CheckRDBE", "CheckMark6", "CheckMCI", "MountMark6",
        "CheckTiming", "InitializePointing", "SetModeAtten", "CheckRDBEs", "CheckPointing", "TestRecording"]

    [sequences.StartExperiment]
        name = "Session start"
        description = "Procedures to be carried out at the start of the session."
        repeating = false
        repeat_gap_min = 0
        tasks = ["StartMulticastLogging", "SendReadyMessage", "StartSched", "SendStartMessage"]
```

Coding the tasks

- Use code templates to build new tasks

```
[tasks.StartFSLogFile]
    name = "Start FS log file"
    description = '''
```

```
class StartFSLogFile(Task):
    def add_data(self, log: str):
        self.log_name = log

    def act(self):
        """
        Tell the FS to open a log file

        1. Reads the current master schedule and determines the next or current experiment involving this station
        2. Prompts the user to check this is the log file to write, gives the option to change it
        3. Tells the FS to open the log file and reports the result

        :return: status: True if log file name was set, False if not or an error
        :rtype: bool
        :return: message: text describing success or failure
        :rtype: strings
        """

        # Read the master schedule and get the current or next session code for this site

        self.ok, self.message, sched_name = self.get_next_session()
        if not self.ok:
            # return with a failure if we couldn't get a sched file name
            return

        log_name = "{}{}".format(sched_name, self.config["Station"]["name"].lower())
        fsc = open_FSCommand(self)
```