

# Interrupting Scheduled Observations

The `interruptObs.sh` script operates in one of two modes: focused and global. In focused mode, it will take an input schedule, determine the relevant launched instances for that schedule, and specifically shut down all related screens, tmuxes and processes. In global mode, the script will kill anything related to our scheduling scripts.

See [date\(1\)](#) (-d input) for further details on accepted time stamps. We also support "next" in focused mode, which will parse the input file, find the next gap between observations, and perform the switch then to minimise downtime.

## Focused Mode

In focused mode, you must provide at least an interruption time (in a format accepted by `date`), and a reference schedule to analyse. You can optionally provide a third parameter with a replacement schedule if you do not want to manually staged and run another schedule afterwards. These are all valid examples for killing a specified `./schedule/my_sched.txt` schedule file.

```
lofar@LGC:~/workdir$ interruptObs.sh 2023-06-01T07:00:00 ./myschedule.txt
lofar@LGC:~/workdir$ interruptObs.sh now ./myschedule.txt
lofar@LGC:~/workdir$ interruptObs.sh now ~/scheduling/schedules/staged/sched_2023-06-01.txt
lofar@LGC:~/workdir$ interruptObs.sh next ~/scheduling/schedules/staged/sched_2023-06-01.txt
schedule_replacement.txt
```

## Global Mode

In global mode, you just need to specify a time and the "KILLALL" keyword as the schedule. After this, it will find any running scripts, screens, tmuxes, etc, on both the recording node, LCU, and local node that are associated with scheduling and kill them. This should typically only be used in the case that something has gone horribly wrong.

```
lofar@LGC:~/workdir$ interruptObs.sh 2023-06-01T07:00:00 KILLALL
lofar@LGC:~/workdir$ interruptObs.sh now KILLALL
```

---

Revision #3

Created 30 May 2023 15:26:15 by David

Updated 16 January 2024 14:29:01 by Evan Keane