

(Archived) Executing a Schedule and Processing Data

Running Observations

Automatic Configuration on the LGC

The observing process has been simplified to only require the use of a schedule file and a single command on the `LGC`. The current working directory can be accessed via the `ilofar` used on the LGC and running the alias `lsched` in the console. The current directory is

```
ilofar@LGC:~/Scripts/Python/sched_bf/tmp_drockennn.
```

Once in the directory, you can create your schedule file as described above, and execute the `scheduleAndRun.sh` with an input file name to automate transferring the scripts to the `LCU` and `LCC1`, and setup the beamforming and recording processes as needed. This command does not need to be run in a `screen/tmux` and will exit early if there is an issue with the input schedule file. Issues can be debugged using the `quicksched.py` script and the `--no-write` flag.

```
bash scheduleAndRun.sh sched_YYYY-MM-DD.txt

# Run into issues? Run the quicksched.py script separately to debug the file
python3 quicksched.py sched_YYYY-MM-DD.txt --no-write
```

Manually Configuring the LCU

In order to perform your observation, you will need to transfer the `lcu_script_YYYYMMDD.sh` script to the LCU. In the case that handover has already been performed, you can simply transfer the script to the station (preferably a sub-directory in `~/local/scripts`) and run it in a screen as a normal bash script. Starting from the LGC on the `ilofar` account, you can use the `lcc` alias to open an SSH connect to the station, then you should navigate to the `~/local/scripts` folder and

either use, or create, your own sub-folder for your observations.

```
ilofar@lgc ~ $ lcu
user1@lcu ~ $ cd local_scripts/
user1@lcu ~/local_scripts $ mkdir my_dir; cd my_dir
```

You may need to find a copy of `sleepuntil.sh` in the parent folder if you are creating a new directory. Afterwards, you can open a screen and execute the script.

```
lcu ~/local_scripts/dmckenna $ screen -S dmckennaObs
lcu ~/local_scripts/dmckenna $ bash lcu_script_YYYYMMDD.sh
< Control a, b to exit >
```

Scheduling before local mode

However if we do not yet have the station and the observation will start before you are available for the day, there is a script on the LGC at `~/David/transfer.sh` that can take an input and transfer it to the station, then launch a screen and execute the script unattended.

The `transfer.sh` script depends on the `sleepuntil.sh` script located in the same folder, if you make a copy of `transfer.sh` to modify the target directory, be sure to copy `sleepuntil.sh` to the same folder to ensure it can run.

The `transfer.sh` script takes 3 inputs: the path of the script to transfer, the date to transfer (as YYYYMMDD) and the time to start trying to transfer (HHMMSS). As an example, to transfer a script for handover occurring at 2022-02-02T08:00:00 the following command could be used to start trying to transfer file 30 minutes prior to handover. In the case that we receive the station early, this will allow for the file to be transferred and station to be configured as soon as possible.

```
ilofar@LGC ~/David $ tmux new -s transferLCuScript
< enter tmux shell>
ilofar@LGC ~/David $ bash transfer.sh lcu_script_name.sh 20220202 073000
< Control d, b to exit >
```

Recording on UCC1

Given `ucc1` should always be available, the script can be transferred to the recording drive and executed as needed. We currently perform all recording on the `ucc1_recording2` drive which can be easily accessed via the `cd2` alias on the `obs` account. After that, you can get the required scripts in the scripts folder, and make a sub-directory for the given source category and perform the observation.

```

# Connect to ucc1
ilofar@LGC ~ $ ucc1

# Navigate to the recording directory
obs@ucc1 ~ $ cd r2

# Make your recording folder
obs@ucc1 /mnt/ucc1_recording2/data $ mkdir rrats/2022_02_02

# Copy the recording scripts to the folder
obs@ucc1 /mnt/ucc1_recording2/data $ cp ./scripts/* ./rrats/2022_02_02/

# Enter the folder
obs@ucc1 /mnt/ucc1_recording2/data $ cd rrats/2022_02_02

# Add your ucc_script_YYYYMMDD.sh script to the folder via rsync, scp, nano + copypaste, etc

# Open a new tmux shell
obs@ucc1 /mnt/ucc1_recording2/data/rrats/2022_02_02 $ tmux new -s recording
<attach to tmux shell>

# Run the observing script
obs@ucc1 /mnt/ucc1_recording2/data/rrats/2022_02_02 $ bash ucc_script_YYYYMMDD.sh

# Close the tmux shell after all observation have been completed.

```

Processing Observations (TODO)

These instructions assume you are in the standard processing Docker container. This can be accessed using the following command, after which you can return to your previous directory with the `$LAST_DIR` variable.

```

obs@uccN ~ $ docker run --rm -it -e TERM -v /mnt:/mnt --gpus all --env LAST_DIR=$(pwd) pulsar-gpu-dsp2021
root@AAAAA /home/soft $ cd $LAST_DIR

```

Standard Observations (TODO)

Rough overview: use `lofar_udp_extractor` to process the input to the desired output format.

```

$ lofar_udp_extractor -i /path/to/input/file/udp_1613%d_2022-02-02T02:22:22.000.zst -p "$PROCESSING_MODE" -o "$OUTPUT_FILENAME" -a "$HEADER_PARAMETERS" -d "0,0,COORDS" -c "ANT,SBB:SBB"

```

Channelising Voltages with Digifil (TODO)

Rough overview: Produce a dada-compatible output with `lofar_udp_extractor`, copy a header to the same directory, process it with `digifil`

```
$ lofar_udp_extractor -i /path/to/input/file/udp_1613%d_2022-02-02T02:22:22.000.zst -p 10 -o  
"$INPUT_FILENAME".dada  
$ cp generic.hdr "$INPUT_FILENAME".hdr  
$ digifil -F "$NUM_CHANS":1 -d "$N_POL_OUT" -i 0 -c -b 32 -B 512 -t "$T_DOWNSAMPLE_FAC" -o  
"$OUTPUT_FILENAME".fil "$INPUT_FILENAME".dada
```

Pulsar Observations

Pulsar observations are currently a Rube Goldberg machine to generate coherently dedisperse filterbanks with a factor of 8 channelisation via `cdmt`, wrapped with a python script called `cdmtProc.py`. This guide assumes that you are observing a source in mode 5, using subbands 12:499, using the standard recording scripts to produce an output with 8x channelisation and 655us sampling.

This process assumes that your source is available in `psrcat`, or has an entry in the `dmFallback.txt` file in `/mnt/ucc3_data1/data/mdrs/` folder, and that a fake sigproc header has been generated and provided in `/mnt/ucc3_data1/data/mdrs/mode5r/`. If the source is available in `psrcat`, the command to generate a fake header will be generated in the output from `cdmtProc.py`, but you will need to transfer this header to `ucc3` manually.

The latest version of `cdmtProc.py` is found at `/mnt/ucc4_data2/data/David/`, and is run using the following syntax, where the parent folder is the main folder used for recording, such as `/mnt/ucc1_recording2/data/mats/2022_02_22/`. Afterwards, a script produces an output `cdmtProc.sh` which can be run to generate both 32-bit and 8-bit filterbank files which can then be processed as needed.

```
root@AAAAAAA /mnt/ucc4_data2/data/my_processing_dir # python ../cdmtProc.py -i  
/mnt/ucc1_recording2/data/route/to/parent/folder/  
root@AAAAAAA /mnt/ucc4_data2/data/my_processing_dir # bash cdmtProc.sh
```

Revision #1

Created 30 May 2023 15:00:26 by David

Updated 30 May 2023 15:01:14 by David