Espresso

Espresso is a suite of scripts designed to take the drudgery out of file-based correlation. The idea is to automate as much of the processing as possible. While providing a rapid way to set up and run correlation jobs, the scripts assume the user is 'intelligent' (that's another way of saying the scripts are dumb) - check their output. All the scripts will give some help if initiated with the `–help` switch. Please read those help messages before using. A typical Espresso correlation would look like this:

```
disk_report.py > ~/disk.json
```

creates a summary of the disks in $DIFX_MACHINES used for storing baseband data.

Transfer the data to the correlator, e.g.:

```
ssh rey075@cass-01-per.it.csiro.au
globus-url-copy <options> -r file://<path>/v255z/ATCA/
sshftp://user@magnus-data/<path>/v255z-At/
```

Following the naming convention for the directories on the correlator (<expname>-<Telescope>) is important.

This transfer can be automated using the `make_copyfiles.py` and `run_copyfiles.sh` scripts on cass-01-per:

```
ssh rey075@cass-01-per.it.csiro.au
scp cormac@magnus:disk.json ~          #be sure to get a newly generated
version of disk.json
cd ~/transfers/v255z
dtsum v255z     # alias for du -cs -BG > du.txt
make_copyfiles.py ~/disk.json ./du.txt
run_copyfiles.sh copy_v255z_*.sh
```

`make_copyfiles.py` will identify destination nodes with sufficient space and create a bunch of shell scripts with the relevant transfer commands. The `run_copyfiles.sh` command can then be used to initiate these transfers under `screen`. Use `screen -r` to view the running transfers.

Update the disk report once the data have been transferred.

```
disk_report.py > ~/disk.json
```

Get the .key file from ATNF and run through our local version of SCHED if necessary. This is no longer generally needed unless the vex file needs modification.

```
cd $CORR_HOME/2015/September/v255z/sched
getatnf v255/v255z/v255z.key             # alias for wget
ftp://ftp.atnf.csiro.au/pub/people/vlbi/v255/v255z/v255z.key
sched < v255z.key
cp v255z.vex ../
```

```
cd ../
```

Also may need certain setup files from ATNF if these are not available locally.

~~You may also have to edit the TRACKS section in the vex file to add the following line:~~
~~S2_recording_mode = <samprate>x<nchan>-<nbits>;~~
~~e.g. for 32 Msample/sec recording, 4 channels, 2 bits, this would be:~~
~~S2_recording_mode = 32×4-2;~~

If an ATCA pad other than the default W104 was used, then update the station position. Can check this on the experiment wiki page under 'ATCA antenna summary' or by checking the refant at the start time of the experiment with:

```
atcapos.py 2015y274d20h00m00s
```

Similarly update ASKAP position if necessary. If Tid used a station other than what was specified in the vex file, it is probably best to re-run SCHED. However you could use `updatepos.py` to swap between two 34m antennas as these have the same slewing parameters etc.

```
updatepos.py ATCA AT_W102 v255z.vex
```

Create an input (.datafiles) file for lbafilecheck.py:

```
disk_exper.py v255z ~/disk.json
```

Create filelists, machines, threads and run file with lbafilecheck.py:

```
lbafilecheck.py v255z.datafiles
```

Edit a .v2d file as required for your experiment. (Note: getEOP.py to download the latest EOPs.)

```
getEOP.py <date> >> v255z.v2d
```

where <date> is the start date of the experiment in MJD or VEX format.

Run vex2difx and check that number of jobs, etc. look reasonable and .v2d and .vex files parse correctly.

```
vex2difx v255z.v2d
```

Start a clock search job (output will go `$CORR_DATA/<expname>/clocks`). Choose appropriate mjdStart and mjdStop, scan or source selection in the .v2d file.

```
espresso.py -c v255z_1
```

Set up the pipeline directories if they don't exist yet; can use the bash alias: `mkpipe v255z`
Run the LBA AIPS pipeline to get clock estimates (either use `mkpipe` or copy a template .inp file from ~cormac/progs/Pypeline/lba_template.inp and edit appropriately).

```
cd $CORR_DATA/v255z/clocks
```

```
fitswrap "v255z_1 V255Z.CLK.FITS"
lbawrap $PIPE/v255z/in/v255z.inp
```

`fitswrap` and `lbawrap` are SLURM wrappers for difx2fits and LBA.py respectively to run the commands on the batch scheduler.

The fitted clock residuals will be in `$PIPE/v255z/out/v255z.clocks`. Or you can see the raw data graphically in `$PIPE/v255z/out/v255z_FRING_DELAY.pdf` and `v255z_FRING_RATE.pdf`.

If you need to inspect the FRING output, e.g. to determine the time of a clock jump, you can use the Pypeline script:

```
clocks.py -p "1000 V252AZ UVDATA 1 1"
```

where 1000 is the AIPS user number, and “V252AZ UVDATA 1 1” is the AIPS name, class, disk and sequence number.

Update the clocks in the .v2d file:

```
cd $CORR_HOME/2015/September/v255z
updateclock.py -e 2012y067d04h00m00s -o 'AT=0.5,HO=-.01,MP=0.034' -r
'AT=5.1,HO=-3.3,MP=1.2' -f 6642 v255z.v2d
```

Units for the clock offset (-o) are microsec, the clock rate (-r) mHz, and the observing frequency (-f) MHz.

If this experiment requires multiple correlation passes, then make copies of the .v2d file for each pass, e.g.: `v255z-line.v2d` and `v255z-cal.v2d`. The naming convention `<expname>-<passid>.v2d` must be followed.

Run the production correlation job

```
espresso.py -a v255z-cal
```

Go make yourself an espresso. You may have time for more than one.

Occasionally jobs will fail to complete. `espresso.py` will list jobs that do not have the usual 'BYE' messages in their logs at completion. `chklogs` is a bash alias which can also check which jobs did not appear to complete as expected, based on the messages at the end of the *.difxlog files. Specify the output data directory, or by default `chklogs` uses the current working directory.

Run the pipeline on the complete dataset, and transfer the pipeline data and summary to the wiki:

```
cd $CORR_DATA/v255z
fitswrap "v255z-cal_?.difx V255Z.FITS"
lbawrap "$PIPE/v255z/v255z-cal.inp"
cd $PIPE/v255z/out
lba_feedback.py v255z-cal.wikilog v255z-line.wikilog > v255z.txt
archpipe v255z
```

Notes:

- Generally may want to set options/output file name in `difx2fits` (run with no options for help). Especially important for generating output from correlation with multiple phase centres (option –`phasecenter <p>`).
- lba_feedback.py can now take multiple .wikilogs, e.g. for different correlator/pipeline passes. Naming convention is {expt}-{pass}.

Archive the correlator output directory using the `archivec.py` script to access the Pawsey data store.

```
archivec.py $CORR_DATA/v255z/ /projects/VLBI/Archive/LBA/v255
```

See the archive page for notes on distribution of output data to PIs.

~~On cortex, link the .FITS files from the archive to the ftp site so the PI can download them: <code> ssh cormac@cortex.ivec.org ln /pbstore/groupfs/astrotmp/as03/VLBI/Archive/Curtin/v255/v255q/*.FITS /pbstore/groupfs/astrotmp/astronomy/anonymous/v255q </code>~~ ~~**Note:**~~ ~~cortex.ivec.org has a special “feature” that makes it necessary to make a soft link in one's home area to the /pbstore partition for transfers to work: <code> ssh cormac@cortex.ivec.org ln -s /pbstore ~/pbstore </code>~~