Setting up for correlation

For DiFX1.5.* and higher. (Notes on the using the old CIRA-DiFX-1.0 are here if required)

The basic steps in running the new version of DiFX are:

1. vex2difx - takes .v2d file as input and produces .input and .calc files (also .flag)
2. calcif2 - takes .calc file as input and produces model files (for DiFX 1.5 .uvw, .delay, [.rate, .im also produced])
3. mpifxcorr - writes output files to .difx directory (as specified in input file)
4. difx2fits - converts output to FITS (and ascii tables including VLBA-style “sniffer” data)

See also difx_run for generic instructions

Espresso: a few local scripts that will make your life easier (see below for usage):

- disk_report.py - summary of disk space usage on the data areas defined in `$DIFX_MACHINES`
- grepexp.py - interrogate the output of disk_report.py and summarise the data for a given experiment
- disk_exper.py - will produce a default input file for lbafilecheck.py based on the output of disk_report.py
- lbafilecheck.py - creates file lists, machines, run and thread files
- getEOP.py - gets EOPS from IERS and returns them in v2d format
- updatepos.py - update a station position in the vex file with a position from `$STADB`
- atcapos.py - find out which ATCA pad was the refant at any given time
- updateclock.py - update clock information in the .v2d file
- mjd2vex.py - converts between the various DiFX date formats (vex, MJD, ISO8601, VLBA).
- espresso.py - wrapper for vex2difx, calcif2, errormon2, mpifxcorr and moves various files to the data areas to facilitate fits file creation and archiving.
- LBA.py - an AIPS (ParselTongue) pipeline for clock searching and data verification.
- archivec.py - archive the final data to the Pawsey data store.

A typical Espresso correlation would look like this.

Preliminaries

Useful links

Observers wiki at ATNF

Data tracking spreadsheet

Status of e-transfers from Warkworth

Curtin LBA correlator records spreadsheet. To be kept up to date as experiments are observed/transferred/correlated.

Notes

- The contact author should be emailed regarding their desired correlation parameters. For pulsar gating/binning, an ephemeris is needed in the form of a `polyco` file in tempo 1 format. See this VLBA memo by Walter Brisken for more information.
- All data from the experiment need to be transferred to /scratch on magnus-data for correlation. If any appear to be missing, check the status at the links provided above. See the espresso example page for usage of scripts to facilitate transfers to Magnus.
- Also need to know which ATCA station pad was used as the reference for the tied array (vex default is W104, used most often). Chris Phillips wrote a script to extract the relevant information which is linked from the observer's wiki for each experiment; Refant is the tied array reference antenna: Example usage. `atcapos.py` provided with espresso will also return the reference antenna for any given date/time.
- If Tid was used, which antenna(s)? It's best to re-run SCHED with the correct one if necessary. If Tid used more than one antenna, a bit of fiddling may be required (I usually use a second station code 'Td'). **Note** this is now handled in `$SCHED/catalogs/stations.dat`, although if DSS43 and DSS45 are both used, you may need to create a local copy of `stations.dat` and edit one of the 2-letter codes to be Td.
- Log in to magnus and make a working area for the experiment: `$CORR_HOME/<year>/<session>/<expcode>`
- Keep notes of setup and any problems encountered, in a file called `<expcode>_notes.txt`.
- Lost time or other problems should also be recorded here on the wiki. Each experiment should have a local wiki page for notes on correlation and analysis plots, that also links to the observers wiki (generated by lba_feedback.py). The aim is to make this a "one-stop" point for the PI to obtain all relevant information.
- Get the vex file and (if needed) SCHED keyin file <expcode>.key (and any necessary associated files) from the ATNF ftp area (linked from the observers wiki). ATCA position defaults to pad W104 (updatepos.py can be used to change this). ASKAP position may also need to be updated (currently ASKAP32 is for X-band; ASKAP07 is used for L-band). It is no longer generally necessary to re-run SCHED, as recent releases have up-to-date station information.
- If ATNF stations used dual DAS recording, the 2 DASes should have been merged and converted to mark5b format. The VEX file will need to be updated to reflect this: see `addMark5b.pl`.
- Generally we take source coordinates from the vex file. Occasionally the PI may supply updated coordinates for correlation - these can be added in the .v2d (preferred) or the vex file.

Create the .v2d file

- Currently, easiest way is to manually copy .v2d file from another experiment and edit as necessary.
- See vex2difx documentation for detailed lists of parameters

Important bits

- Set the vex file name in .v2d file.
- Check data format setting in .v2d for LBA stations (usually LBAVSOP for 16 MHz, LBASTD for 64 MHz).
- If desired can override default parameters e.g. for maximum output file size [2 GB], maximum length of job [7200s] by specifying maxSize and maxLength. maxGap may also be increased

(e.g. it may desirable to correlate an entire experiment in 1 job - usually no reason to do this though).

- Correlation parameters are specified in the SETUP section(s) which are referenced by RULE section(s).
- ANTENNA blocks use 2 letter station codes. Can specify name if desired, but two-letter station codes are preferred (then no need to specify the name in the ANTENNA block. I usually make a note of which Tid antennas were used to help the PI, but no longer change the name).
- ~~Include ATCA X,Y,Z corresponding to reference station, and include position for correct Tid antenna, if needed.~~ If needed, e.g. to correct the ATCA reference station or the Tid antenna position, updatepos.py will replace the position in the vexfile with a position from $STADB. (Updating positions in the VEX file is preferred over including station positions in the .v2d file, as vex2difx will automatically take care of antenna motion corrections (assuming dx, dy, dz are available).)
- **Note:** Station positions are taken from the vex file unless they are included in the .v2d file.
- We use filelist keyword in the ANTENNA block to point to the list of data files to correlate (created by `lbafilecheck.py`).
- EOP section + ANTENNA clock offsets can be inserted in .v2d file. (These may also be appended to the vex file.) For EOPs, there is a script `getEOP.py` to get the latest EOPs for 5 days surrounding the specified date from `http://gemini.gsfc.nasa.gov/solve_save/usno_finals.erp` in the format required by vex2difx.
- Zoom bands must be set in the .v2d file if different stations recorded different channel bandwidths. A zoom band must be smaller than (and not equal to) its parent band. However, since DiFX-2.4 a zoom band for one station can match that of a different station.

About EOPs (all OK if using getEOP.py):

- Would typically include in .v2d EOPs for 5 days (i.e. include 2 days either side of observed MJD) from usno_finals.erp. Do not include more than 5 days of EOPs as calcif2 will simply ignore values after the 5th.
- Watch units!
  - Polar motion: .erp uses 0.1 arcsec; .v2d requires arcsec
  - UT1-UTC: .erp uses microsec; .v2d requires offset seconds

File lists

- There is a script 'lbafilecheck.py' that will automatically generate the filelists, check that they have valid headers and append start/stop times for each. It will also create a prototype machines and thread file for mpi, and a prototype run file to start correlation. It assumes that you want to use all the nodes listed in a file pointed to by the environment variable $DIFX_MACHINES for the correlation job. File names must follow a convention whereby an alphabetical listing corresponds to sorting in time order.

```
lbafilecheck.py <expname.datafiles>
```

the format of the file <expname.datafiles> is described here

A default input file for lbafilecheck can be produced using disk_report.py (which summarises the of data areas defined in $DIFX_MACHINES), and disk_exper.py (which reads the output of disk_report.py):

```
disk_report.py > ~/disk.json
disk_exper.py v252o ~/disk.json
```

Potential issues

Pulsar binning

Polyco files produced using tempo 2 (to generate tempo 1 format) often have 'coe' instead of the required '0' for the observatory number. This must be edited e.g. with

```
sed -i "s/coe/  0/g" polyco.dat
```

Other things to watch out for with polycos include:

- roundoff errors in times - DiFX reads the integer MJD from the MJD field, but the time in seconds from the previous field; sometimes the MJD may end in .999999… but the time in seconds will be zero, so the wrong MJD is used. Can be fixed by inserting the rounded-up MJD.
- actual gaps between polyco entry times - the correlation will not run if polycos are not found to cover the necessary timerange, but sometimes the error messages are not helpful and the problem is not obvious. If this problem occurs, a new polyco file will be needed (ask your friendly pulsar astronomer).

Data format for LBA recorders - LBAVSOP for 16 MHz baseband channels, but may be LBASTD (for 64 MHz or 4 MHz recording). This *must* be specified in the .v2d file. If wrong, data will look “normal” but with sqrt(2) loss of sensitivity due to swapped bits.

Mark5B recording

~~Mark5B recording currently usually requires editing of the $TRACKS section of the vex file, as documented here.~~ Mark5B is now handled by SCHED/vex, so in general no fiddling is needed.

`m5bsum` (Mark5 summariser) is a useful program to check the format of the data if unsure.

Missing headers

This is automatically fixed for you if you create the filelists using 'lbafilecheck.py' so no need to worry about this unless you are creating the file lists by hand. It used to be quite common for a few LBA files per experiment to have missing or corrupt header information. Old versions of DiFX don't handle this (DiFX-1.5.3 on CUPPA has been patched for this; later versions should also be o.k.); it will just hang (until killed) when it encounters such a file, hogging CPU cycles. Cormac has written a simple script to check headers for a given list of data files, in `/home/corr/bin/chk_vlbi.pl`. Run with, e.g. `chk_vlbi.pl <station>.filelist` on the appropriate node where the data live (if it is not in an NFS-mounted area).

If a station has recorded with **swapped polarizations**, this can be specified in the ANTENNA section of the .v2d file (boolean parameter polSwap).

It should also be MUCH easier than it used to be to set up for spectral line correlation with the .v2d file 😉

The following may still be an issue, if the vex file does not reflect what was recorded!

- For old schedules the vex file may need editing to reflect some stations not recording all channels. For experiments at 512 Mbps, typically Ceduna and Hobart record half of the channels. (Not recommended, but it is also possible to edit the correlator input file datastream and baseline tables. See the explanation of the correlator input file format.)
- Also noted on the ATNF wiki is what to do in case of band inversion. This is an issue for 64 MHz recording modes. I find it easier to edit the $FREQ block in the vex file! The same fix is needed; just add 64 MHz to the relevant frequency channels and change sideband from U to L. Note that in DiFX-1.5.1, if some stations are inverted, but others are not, then you must hack the .vex or .input to have both the U and L modes for running mpifxcorr, but for calcif2 and difx2fits, you must again hack the .input (or .vex) so that it only refers to one sideband for all stations. DiFX-1.5.3 has a local patch that makes the single-sideband hack for calcif2 and difx2fits unnecessary. This patch should appear in later versions also.

Running correlator jobs

The simplest way to start a correlator job is to use the Espresso shortcut:

```
espresso.py <jobid_1> <jobid_2> ... <jobid_n>
```

It automatically carries out the steps described below, as well as doing a few bookkeeping things like ensuring the data are written to the standard data areas, and backing up old jobs before removing them from the output directory.

It requires that the file lists, run and thread files have prototypes produced by lbafilecheck.py (or similar).

Run vex2difx

```
vex2difx <baseFilename>.v2d
```

Run calcif2

CALC_SERVER should be set to the IP address of a machine running calcserver (ask Cormac or Hayley in case of problems)

```
calcif2 <baseFilename>.calc
```

Set up the correlator run file

lbafilecheck.py will create “run”, “machines” and “threads” files but these may not be optimal for your needs, in which case:

- Create the threads file (to match the name specified in the .input file), machines file, and run

file (copy from another experiment and edit as required).
- Start `errormon2` to log DiFX messages
- Run the correlator!

Clock searching notes

Don't know which source is a good fringe-finder?

Check in the latest catalogue compiled by Leonid Petrov

Ceduna clock can be erratic but is monitored. An historical graph should also be available.

Using AIPS for fringe-finding

The simplest way is to run the LBA pipeline.

If you want to do it by hand:

Start AIPS with `aips tv=local:0`

Use tasks FITLD (ATLOD for RPFITS), POSSM, FRING, SNPLT. Add delays to ANTENNA sections in .v2d file. Calculate long-term clock rates as required. (If necessary, offsets for particular IFs can be inserted in the DATASTREAM table of the generated correlator input file.)

updateclock.py can be used to update the clocks in the .v2d file.

```
updateclock.py -o 'AT=2.2,PA=-0.2' -r 'AT=0,PA=2' -f 8425 test.v2d
```

would adjust the AT clock by 2.2 microsec in delay and 0 in rate, and the PA clock by -0.2 microsec in delay and 2 mHz in rate for a frequency of 8425 MHz.

Verification

Analysis of data in AIPS