DiFX Programs and Utilities

This page briefly explains the purpose of the myriad utilities that have been developed in aid of debugging and operations support. These are organized by topic. For an alphabetical list of these with considerable more detail, visit the Reference Manual.

Note: not all of the tools listed below are getting installed automatically by the install_difx script. Typically configure/make tools exist to install these manually if required.

Core DiFX Programs

- `mpifxcorr` : The software correlator itself
- `vex2difx` : Makes `.input` and other files used by `mpifxcorr` from supplied `.vex` file
- `difx2fits` : Takes the output of `mpifxcorr` and generate `.FITS` files
- `difx2mark4` : Takes the output of `mpifxcorr` and generates Mark4 format files
- `difx2profile` : Takes the output of `mpifxcorr` and averages data of all antennas over the entire time, in preparation of `profile2binconfig.py`. Related to pulsar processing.
- `calcif2` : Takes `.calc` file made by `vex2difx` and produces the delay model (`.im`) file
- `CalcServer` : A service that makes calls to the CALC program (Calc9) upon request by an RPC call, easiest started with `startCalcServer`.
- `difxcalc` : creates .im model (based on Calc11) from .calc files. Replaces the CalcServer.

DiFX Operations

- `difxcopy` : Copies DiFX output to another directory and updates internal path references (e.g. in `.input`) accordingly.
- `difxlog` : Captures multicast status and messages from a running DiFX job and writes to a text file
- `difxsniff` : Concattenates as appropriate output sniffer files from `mpifxcorr` and plots using `plotwt`, `plotapd` and `plotbp`
- `filteredDifx2Fits` : Prepare difx2fits jobs based on source, timerange and/or mode.
- `genmachines` : Determines location of Mark5 modules via request to `mk5daemon` processes and makes `.threads` and `.machines` files
- `geteop.pl` : Script to obtain EOP values from GSFC servers in VEX $EOP format.
- `makefits` : Simplified wrapper interface for `difx2fits`, tailored to NRAO's usage
- `oms2v2d` : Makes skeleton `.v2d` file based on the `sched .oms` output file
- `startdifx` : Command line utility to start `mpifxcorr`
- `stopdifx` : Command line utility to stop a single running instance of `mpifxcorr`
- `vexpeek` : Program that reads a `.vex` file and prints the project name and list of stations and the time period each observed; used by `db2vex`
- `vlog` : Splits up a TSM style text log file into `pcal`, `weather`, `tsys` and `flags` files; mainly for VLBA operations
- `startcorr.pl` : Simple DiFX launcher, with eVLBI support
- `difxwatch` : Monitors progess of running mpifxcorr. Kills job if no progess for longer than 300s.
- `espresso`: a lightweight suite of programs designed for management of file-based correlation.

DiFX File Validation

- `checkmpifxcorr` : Checks the contents of `.input` file, and files referenced within, for syntax compliance
- `difxcalculator` : Uses difxio to load the `.input` file and prints a summary of memory, cpu, and network utilization for a job

DiFX Monitoring

- `cpumon` : Displays multicast information, as usually sent by `mk5daemon`, such as CPU, memory and network utilization, from DiFX cluster members
- `errormon` : Prints multicast messages to screen
- `errormon2` : Similar to `errormon`, perhaps with more useful defaults
- `mk5mon` : Displays multicast information about Mark5 units
- `mk6mon`: Displays multicast information about Mark6 units
- `statemon` : Prints condensed summary of the DiFX state messages

VDIF tools

- `vsum` : Prints summary information about a vdif file. Can produce filelist to be digested by vex2difx
- `vmux` : A program to take a multi-thread VDIF file and multiplex into a multi-channel, single thread file
- `printVDIFheader` : A program to dump some basic info about VDIF packets to the screen
- `printVDIFgaps` : A program to look for missing VDIF packets
- `printVDIF` : A program to dump some basic info about VDIF packets to the screen
- `vdifbstate` : A VDIF state counter for multi-thread VDIF data
- `vdif2to8` : A program to take a VDIF file containing 2-bit samples and convert it to 8-bit samples
- `vdifChanSelect` : A program select a subset of channels from a VDIF file
- `vdifd` : A VDIF decoder for multi-thread, single channel VDIF data
- `vdiffold`: A VDIF decoder for multi-thread VDIF data
- `vdifheader.pl` : A program to dump some basic info about VDIF packets to the screen
- `vdifspec` : A VDIF spectrometer for multi-thread VDIF data
- `vdif_time` : ?
- `cleanVDIF` : Walks packet-by-packet through a VDIF file looking for packets of invalid length (often ARP packets from a switch inadvertently captured by a Mark5C) or corrupted header info, stripping out bad packets and writing a 'clean' VDIF file. Fix Me!: document in user guide
- `multi2singlethreadVDIF` : Takes a VDIF file with 2^N single-channel threads and multiplexes these channels into a single VDIF thread with multiple channels. The resultant single thread output file can be processed by e.g. m5spec and other utilities, and by mpifxcorr. Fix Me!: document in user guide
- `padVDIF` : Walks packet-by-packet through a VDIF file looking for missing packets and inserting packets as necessary to make a continuous packet stream. The inserted packets have the invalid bit set. Fix Me!: document in user guide
- `printVDIF` : Loops through a VDIF file inspecting each packet header and printing some basic summary info (time etc)
- `stripVDIF` : Strips UDP overhead from VDIF packets captured using wireshark, leaving a correct VDIF file. Fix Me!: document in user guide

Mark5 data tools

- `m5bstate` : Mark5 state counter that produces sampler statistics.
- `m5d` : Decodes the header and samples of baseband data from a mark5access supported baseband data file
- `m5fold` : Folds in time the voltage squared (akin to pulsar processing); can be used to identify switched power in a datastream
- `m5findformats` : Tries to determine the data format of mark5 data files
- `m5test` : Reads through a baseband data file looking for missing or damaged frame headers
- `m5tsys` : Extracts switched power from a baseband data file (rather, it extracts Pon and Poff for each sampled channel)

Baseband tools

- `zerocorr` : A program that takes two baseband data signals and performs a zero-fringe-rate cross-correlation; can handle mismatched bands/sidebands and different data formats and sample rates
- `m5spec` : Forms spectra of each sampled channel in a baseband data file
- `m5subband` : Extracts via filtering an arbitrary narrow subband out a wideband recording.

Phasecal tools

- `m5pcal` : Extracts pulse cals from a baseband data file
- `plotpcal` : Versatile P-Cal tone plotting utility, with multiband delay estimation.
- `plotpcal2` : ?
- `plotDiFXPCal.py` : Fast P-Cal tone plotting utility, with multiband delay estimation within and accross bands, PDF export.

Pre correlation tools

- `geteop.pl` : Obtain EOP parameters from usno_finals.erp
- `fslog2difx.pl` : extract DiFX relevant information from the Field System station logs

Post correlation tools

- `pcList.pl` : Determines FITS completeness by compare the FITS file contents against the vex files specifications
- `plotDiFX.py` : Given one or more .difx output files, plots amplitude, phase and lag information to the screen (overplotting in different colours when >1 file given) one subband at a time. Loops through the records sequentially. Fix Me!: document in user guide
- `plotDynamicSpectrum.py` : Given a .difx file, makes a dynamic spectrum of the visibilities (either a single baseline or scalar averaged over all baselines). Fix Me!: document in user guide
- `printDiFX` : Prints summary visibility information of DiFX output files
- `polswapDiFX.py` : Swap polarization labels in DiFX .difx output file for specified station(s).
- `difx2difx.py` : Take DiFX output and un-zoom selected zoombands into new contiguous

bands

DiFX Testing

- `diffDiFX.py` : Generates a context-sensitive difference of two DiFX output files for detailed version testing. Corresponding visibility records are differenced and statistics on the differences are accumulated and printed at the end of the processing.
- `neuteredmpifxcorr` : Runs mpifxcorr without actually writing out visibilities. Fix Me!: document in user guide
- `snipDiFX.py` : Given an input .difx file and a timerange, extract only those visibilities falling in the specified timerange and copy them to a new .difx file. Fix Me!: document in user guide
- `testdifxinput` : Prints a summary of a `.input` file based on the perspective of the difxio library
- `testdifxmessagereceive` : Prints details of multicast DiFX messages
- `vlbi_fake` : A program to benchmark the performance of sending data around the network Fix Me!: document in user guide

Mark6 Operations

- `fuseMk6` : Fuse mount Mark6 modules with recordings in any format.
- `vdifuse` : Fuse mount Mark6 modules with recordings in VDIF format, with VDIF inspection.
- `mk6mon`: Displays multicast information about Mark6 units
- `mk5daemon` : If invoked with -6 option will handle Mark6 operations e.g. module mounting on keyturn etc.
- `m6sg_mount` : (Un)mount any inserted Mark6 modules
- `scan_check` : Examine a mark6 file (either sg fragments or fused-VDIF) and report issues

Mark5 Operations

- `condition` : Queries database for results of module conditioning; also allows injection of new conditioning results
- `condition_watch` : A service that listens for multicast conditioning information and adds new records to database
- `mk5cat` : Open for read a scan or byte range on a Mark5 module and send to stdout; there seems to be some minor issues left with this (work in progress)…
- `mk5control` : Utility to send a command to one or more `mk5daemon` processes
- `mk5cp` : Copies one or more scans or a byte range from a Mark5 module to one or more files
- `mk5daemon` : A process that runs on all DiFX cluster members, not just Mark5 units, that logs and multicasts status and is responsible for starting DiFX jobs and fielding commands sent by `mk5control`
- `mk5dir` : Reads the user directory from a Mark5 module and stores it in a file; by default this reads a bit of data from the beginning and end of each scan and is thus useful for identifying malformed data
- `mk5erase` : Erases a Mark5 module; like the Haystack `SSErase` program but with more options; can set the directory to be either legacy or new format and transmits progress via multicast messages
- `recover` : Like the `Mark5A` recover command
- `testmod` : Does read and/or write tests of Mark5 modules

- `vsn` : Retrieves or sets the Volume Serial Number of a Mark5 module; also prints the serial numbers, model numbers and capacities of the drives in the module.

HOPS related

- `difx2mark4` : convert .difx correlator output to the mark4 output format digested by the HOPS programs
- `vex2ovex` : convert a .vex file to .ovex which is needed by some utilities of the HOPS suite of programs.

NRAO specific

- `e2ecopy` : NRAO specific program to move, rename, and set permissions of files bound for the VLBA archive
- `getvex` : Copies `.vex` file and other bits from a standard filesystem location within NRAO operations

DiFX Database Tools (NRAO specific)

- `difxarch` : Calls `e2ecopy` to transfer correlator output to a final archive destination, consulting a list of expected files along the way
- `difxclean` : After archiving of data is complete, removes all DiFX input and output files from the correlation staging area
- `difxqueue` : Copies `.input` and associated files (with `difxcopy`) to a staging area and updates a database of jobs to run
- `difxusage` : Queries database for total correlator usage over a requested time period
- `db2vex` : Database utility that reads a `.vex` file, queries an operations database, and makes a new `.vex` file containing additional information such as clock offsets, earth orientation parameters and Mark5 module VSNs
- `getshelf` : Requests the current shelf location of a module from the operations database
- `jobdisks` : Looks at a `.input` file for the list of Mark5 module VSNs
- `joblist` : Lists all the jobs in the current directory and summarizes antennas and bands to be correlated
- `jobstatus` : Broken for DiFX 2.0
- `listcpus` : Uses ssh to determine CPU type of all machines in a DiFX cluster
- `makefits` : Simplified wrapper interface for `difx2fits`, tailored to NRAO's usage

Libraries for developers

* Python module `difxdb` : ... * Python module `difxfile` : ... * Python module `difxutil` : ... * Python module `mark5access` : ... * C library `mark5access` : decode VDIF and older raw data formats

Other

- `dedisperse_difx` : Fix Me!: document in user guide