VLBI Level 1A SWIN format

Most of this document is written by Walter Brisken, compiled by Leonid Petrov

2/22/22

Abstract

Very long baseline interferometry (VLBI) data analysis chain starts from processing time series of digitized voltages from receivers at observing stations. Voltage records that are considered Level 0 data are processed with a correlator. The correlator takes as input records with voltage, field system log, experiment schedule, and control file and produce the output: time series of cross- and auto- correlation spectra also known as visibility data, amplitudes and phases of phase calibration data and auxiliary information. The correlator output is considered as Level 1A data product.

This document describes the output of the software correlator DiFX (Deller et al., 2007, 2011). Since the correlator was originally developed in Swinburne University, the output format is called SWIN. VLBI Level 1A data in SWIN format consists of a 100–10,000 files depending on an experiment. The document describes file naming convention, file contents, and format.

Contents

1 Contents of a VLBI L1A archive file in SWIN format

A file in VLBI L1A SWIN format is an archive created by program tar that may be compressed with bzip2 utility. That file is specififc to DiFX software. An output from other correlators is totoally different. The achieve has a number of files of a different type that follow a certain naming convention. Some data file types are mandatory, i.e. are present in every archive, some data file types are optional.

1.1 Directory structure of a VLBI L1A SWIN archive

The VLBI L1A SWIN archive has a two-level directory structure. The files are created by DiFX, by an analyst, and by the submission process. A VLBI experiment has a nominal start and stop date. Word *nominal* implies that the range of usable data may be less than the nominal range. Two or more observing stations participate in a given experiment. VLBI recording is split into data chunk called scans. A scan has a start and stop date. Two or more stations point to a specific direction during a scan. It may happen that only a subset of stations participate in a given scan. A correlator may process one or more phase centers associated with a target source during a given scan, although in most of the experiments only one phase center is processed. Therefore, a scan may have one or more phase centers for the same antenna pointing. The correlator produces an output for each phase center.

The table below shows the structure of the archive VLBI L1A in SWIN format:

Flag	File name	Contents
M	**xxxxx**_meta.txt	VLBI experiment meta data
M	**xxxxx**.vex	VLBI schedule used for correlation
M	**xxxxx**.v2d	DiFX experiment control file
M	**yyyyyy**_**N**.input	DiFX phase center control file
M	**yyyyyy**_**N**.im	Correlator interferometric model
M	**yyyyyy**_**N**.calc	Parameters of the correlator model computation
O	**yyyyyy**_**N**.filelist	the list of data files to correlate
O	**yyyyyy**_**N**.joblist	the list of DiFX jobs to run
O	**yyyyyy**_**N**.difxlog	DiFX alert and DiFX status messages
O	**yyyyyy**_**N**.flag	the list of data to exclude from correlation
O	**yyyyyy**_**N**.machines	the list of nodes to run DiFX
O	**yyyyyy**_**N**.threads	The list of threads to run DiFX
M	**yyyyyy**_**N**.difx/DIFX_**DDDDD**_**TTTTTT**.s0000.b0000	Complex visibility data
M	**yyyyyy**_**N**.difx/PCAL_**DDDDD**_**TTTTTT**_**SS**	Complex phase calibration data

Flag M stands for mandatory and O for optional. Optional file types may be absent in a given SWIN data archive.

Notation: **xxxxx** is a lower case experiment name as defined in the IVS master file. Experiment name length is in a range of 4 to 7 characters. **yyyyy** is a string of the name base that is common for the entire experiment archive. The string is selected by an analyst and may be arbitrary. **N** is an integer phase center index that runs in a range of $[N_{\min}, N_{\max}]$. If only one phase center was correlated for every scan, the total number of phase centers is the same as the total number of scans. Therefore, there are $N_{\max} - N_{\min} + 1$ files with name in a form **yyyyyy**_**N**.input. For example, if **yyyyyy** is `h` in a given VLBI experiment, $N_{\min} = 1000$, $N_{\max} = 1003$, there will be four files with extension .input: `h_1000.input, h_1001.input, h_1002.input,` and `h_1003.input`. **DDDDD** is an integer modified Julian date at the midnight preceeding a given scan and **TTTTTT** is a six character long nominal scan start UTC time tag in integer seconds from midnight with heading zeros. **SS** is a two character long station code. If K stations participated in scan **yyyyyy**_**N**, then there are K files PCAL_**DDDDD**_**TTTTTT**_**SS** in a given directory **yyyyyy**_**N**.difx. For instance if three stations, MG, WS, and OE participated in scan h_1002 that has a nominal start 2022 February 01 at 19 hours 23 minutes 13 seconds UTC, then there will be three files `h_1002/PCAL_59611_069793_MG`, `h_1002/PCAL_59611_069793_WS`, and `h_1002/PCAL_59611_069793_OE`.

A valid archive may contain other files not described in this documents. This files may be added by analyst or may be a result of a test, or a result of data analysis. Analysis software that utilizes VLBI Level 1A data should not rely on these files.

1.2 xxxxx_meta.txt

A meta file is generated by the data submission software and stores results of parsing the directory with the output from the DiFX correlator.

A file in VLBI Level 1A SWIN meta format consists of records of variable format in plain ascii coding. The first line, so-called UNIX magic, identifies the format:

```
# SWIN-ARCHIVE meta data.  Format version  1.01  2020.11.11
```

Lines that start with # characters are considered comments. Records of the meta file are in a format `keyword:  value`. Keyword and value are separated by one or more blanks. Value may be one word or a blank separated list. Lines have the following order:

Contents of a SWIN metadata file:

exper_name	defines the IVS experiment code. Experiment codes are limited to 7 characters.
corr_vers	defines the correlator version. The value is a positive integer number.
exper_desc	defines the experiment description. If the Level 1A SWIN file has no experiment description ?? is used.
pi_name	defines the principal experiment name or the institution.
corr_name	defines the correlator name.
date_start	defines the nominal start date. The value has four words. The first word is an integer Modified Julian Day on the midnight, the second word is the start time tag in UTC as an integer number, and the third word is the start date string in format YYYY.MM.DD_hh:mm:ss. The fourth word is the time zone, always UTC.
date_stop	defines the nominal stop date. The value has four words. The first word is an integer Modified Julian Day on the midnight, the second word is the stop time tag in UTC as an integer number, and the third word is the stop date string in format YYYY.MM.DD_hh:mm:ss. The fourth word is the time zone, always UTC.
duration	defines VLBI experiment nominal duration. The first word is duration in seconds as an integer number. The second word is unit, always sec.
num_sta	defines the number of stations that nominally participated in the experiment.
num_sou	defines the number of sources that nominally observed in the experiment.
stations	defines the observing station codes. The number of words is equal to `num_sta` value. Station codes are two character long and in the upper case.
sources	defines the list of sources that have been observed. The number of sources is equal to `num_sou` value.
polariz	defines the polarization of recieved emission. The number of words is either one or two. Supported polarization labels: R, L, X, Y, H, V. NB: an early version of DiFX did not support H and V polarizations and incorrectly labeled H polarization as X and V polarization as V.
num_inps	defines the number of so-called DiFX input files. This number is equal to the total number of phase centers. In a case if correlation was performed only for one phase center for each scan, `num_inps` is equal to the total number of scans.
num_files	defines the total number of files in the L1A SWIN dataset.
vex_file	defines the name of the VLBI schedule file in vex format that was used for correlation.

v2d_file defines the name of the configuration DiFX file used during processing of a given experiment.

file defines a file name in the L1A SWIN dataset. The is the only keyword that is defined more than once. The order of `file` keywords follows the order of files in the tar archive.

1.2.1 Fast extraction of the file with metadata

The archive files is created in such a way that files are follow in the this order:

xxxxx_meta.txt
xxxxx.vex
xxxxx.v2d
yyyyyy_N.input

A compressed VLBI Level 1A data file may be very large, over 100Gb, and decompression of an entire file may take hours. Because a meta file is always the first file in the tar archive, it contents can be printed to stdout for a fraction of a second using this UNIX command *without decompression of the entire archive file*:

```
cat FFFFFF | bzip2 -d - | \
    tar --occurrence=1 -xf - ‘cat FFFFFF | \
    bzip2 -d - | \
    awk ’BEGIN { RS = "\0" } ; {print $1} NR=1{exit}’‘ --to-stdout
```

where *FFFFFF* is the compressed tar file.

1.3 yyyyyy_N.vex

The vex (Vlbi EXperiment) file format (Himwich, 2021) is a widely adopted observation description format used for scheduling VLBI observations and for driving the correlation thereof. The original vex file for an experiment is typically created by `sched`, `sur_sked`, VieSched++ or `sked` VLBI scheduling software. This file is used to generate control file for antenna hardware to run observations. It has two major parts. The first part defines an astronomical schedule: the nominal start and stop time, the list of observing stations, and the sequence of blocks for observations called scans: the start time for slewing to a given pointing direction with the specified right ascension and declination, usually associated with a target source, start time for execution of a pre-observation procedure, the start time for recording voltage data, the stop time for recording, and the start time for post-observation procedure. An astronomical schedule is a sequence of scans executed in the specific order. The second part defines the hardware setup: the local oscillator frequency, the number of intermediate frequencies, the intermediate frequency offsets with respect to the local osciallator frequency, polarization channels, the use of hardware for phase calibration generation, etc. A small amount of information based on the actualities of the observation may be added by NRAO program `db2vex`, producing a new vex file with an additional file extension `.obs`. The schedule file used for observations may be renamed to have `orig` in the name. If more than one schedule file in vex format is present in the experiment archive, the file with name yyyyyy_N.vex is the file that was used by the DiFX correlator.

1.4 yyyyyy_N.v2d

The `.v2d` file is used to specify correlation options to `vex2difx` and adjust the way in which it forms DiFX input files based on the `.vex` file. The `.v2d` file consists of a number of global parameters that affect the way that jobs are created and several sections that can customize correlation on a per-source, per mode, or per scan basis. All parameters (those that are global and those that reside inside sections) are specified by a parameter name, the equal sign, and one value, or a comma-separated list of values, that cannot contain whitespace. Whitespace is not required except to keep parameter names, values, and section names separate. All parameter names and values are case sensitive except for source names and antenna names. The # is a comment character; any text after this on a line is ignored.

Most parameters are one of the following types:

- **bool** : A boolean value that can be True or False. Any value starting with `0`, `f`, `F`, or `-` will be considered False and otherwise True.
- **float** : A floating point number. Can be of the forms: `1.23`, `1.2e-4`, `-12.6`, `4`
- **int** : An integer.
- **string** : Any sequence of printable (i.e. non-whitespace) characters. Certain fields require strings of a maximum length or certain form.
- **date** : A date field; see below.
- **array** : Array can be of any of the four above types and are indicated by enclosing brackets, e.g., [int]. The empty list is indicated with [] which is usually implied to be all-inclusive.

All times used in `vex2difx` are in Universal Time and are internally represented as a double precision value. The integer part of this value is the date corresponding to 0^h UT. The fractional part, when multiplied by 86400, gives the number of seconds since 0^h UT. Note that this format does not allow one to specify the actual leap second if one occurs on that day. Several date formats are supported:

- **Modified Julian Day** : A decimal MJD possibly including fractional day. E.g.: `54345.341944`
- **Vex time format** : A string of the form: `2009y245d08h12m24s`
- **VLBA-like format** : A string of the form: `2009SEP02-08:12:24`
- **ISO 8601 format** : A string of the form: `2009-09-02T08:12:24`

Global parameters can be specified one or many per line such as:

```
maxGap = 2000 # seconds
```

or

```
mjdStart = 52342.522 mjdStop=52342.532
```

The following parameter names are recognized:

Name	Type	Units	Defaults	Comments
vex	string			filename of the vex file to process; **this is required**
mjdStart	date		obs. start	discard any scans or partial scans before this time
mjdStop	date		obs. stop	discard any scans or partial scans after this time
break	date			list of MJD date/times where jobs are forced to be broken
minSubarray	int		2	don't make jobs for subarrays with fewer antennas than this
maxGap	float	sec	180	split an observation into multiple jobs if there are correlation gaps longer than this number
tweakIntTime	bool		False	adjust (up to 40%) integration time to ensure it contains an integer number of sub-integrations
singleScan	bool		False	if True, split each scan into its own job
singleSetup	bool		True	if True, allow only one setup per job; True is required for FITS-IDI conversion
maxLength	float	sec	7200	don't allow individual jobs longer than this amount of time
minLength	float	sec	2	don't allow individual jobs shorter than this amount of time
maxSize	float	MB	2000	max FITS-IDI file size to allow
dataBufferFactor	int		32	the `mpifxcorr` DATABUFFERFACTOR parameter
nDataSegments	int		8	the `mpifxcorr` NUMDATASEGMENTS parameter
jobSeries	string			the base filename of `.input` and `.calc` files to be created; defaults to the base name of the `.v2d` file
startSeries	int		20	the default starting number for jobs created
sendLength	float	sec	0.262144	roughly the amount of data to send at a time from datastream processes to core processes
antennas	[string]		[] = all	a comma separated list of antennas to include in correlation
baselines	[string]		[] = all	a comma separated list of baselines to correlate; see below
padScans	bool		True	insert non-correlation scans in recording gaps to prevent `mpifxcorr` from complaining
invalidMask	int		0xFFFF	this bit-field selects which flag conditions are considered when writing flag file: 1=Recording, 2=On source, 4=Job time range, 8=Antenna in job
visBufferLength	int		32	number of visibility buffers to allocate in mpifxcorr
overSamp	int			force all basebands to use the given oversample factor
mode	string		normal	mode of operation; see below
threadsFile	string			overrides the name of the threads file to use
nCore	int			with nThread and machines, cause a `.threads` file to be made
nThreads	int			number of threads per core in `.threads file`
machines	[string]			comma separated list of machines to use as processors first is head node, then datastreams, then cores
maxReadSize	int	bytes	25000000	maximum number of bytes to read at a time
minReadSize	int	bytes	10000000	minimum number of bytes to read at a time

The *baselines* parameter supports the wildcard character `*` an individual antenna name, or lists of antenna names separated by `+` on each side of a hyphen (`-`). Multiple baseline designators can be listed. Examples:

- `A1-A2` : Only correlate one baseline
- `A1-A2, A3-A4` : Correlate 2 baselines
- `*-*` : Correlate all baselines
- `A1-*` *or* `*-A1` : Correlate all baselines to antenna A1
- `A1+A2-*` : Correlate all baselines to antenna A1 or A2
- `A1+A2-A3+A4+A5` : Correlate 6 baselines

A source section can be used to change the properties of an individual source, such as its position or name. In the future this is where multiple correlation centers for a given source will be specified. A source section is enclosed in a pair of curly braces after the keyword SOURCE followed by the name of a source, for example

```
SOURCE 3C273
{
  source parameters go here
}
```

or equivalently

```
SOURCE 3c273 { source parameters go here }
```

Name	Type	Units	Defaults	Comments
ra		J2000		right ascension, e.g., `12h34m12.6s` or `12:34:12.6`
dec		J2000		declination, e.g., `34d12'23.1` or `34:12:23.1`
name	string			new name for source
calCode	char		' '	calibration code, typically `A`, `B`, `C` for calibrators, `G` for a gated pulsar, or blank for normal target
naifFile	string			name of leap seconds file (e.g., `naif0010.tls` for ephemeris
ephemObject	string			name or number of object in ephemeris file
ephemFile	string			path of ephemeris file (either `.bsp` or `.tle` format
doPointingCentre	bool		true	Whether the pointing centre should be correlated (only ever turned off for multi-phase center)
addPhaseCentre	string			contains info on a source to add; see below

To add additional phase centers, add one or more "addPhaseCentre" parameters to the source setup. In the parameter, the RA and dec must be provided. A name and/or calibrator code can be added as well. For example: `addPhaseCentre=name1010-1212/RA10:10:21.1/Dec-12:12:00.34` .

An antenna section allows properties of an individual antenna, such as position, name, or clock/LO offsets, to be adjusted. Note that the "late" convention is used in *clockOffset* and *clockRate*, unlike the "early" convention used in the `.vex` file itself.

Name	Type	Units	Defaults	Comments
name	string			new name to assign to this antenna
polSwap	bool		False	swap the polarizations (i.e., `L` $\Leftrightarrow$ `R`) for this antenna
clockOffset	float	us	vex value	overrides the clock offset value from the vex file
clockRate	float	us/s	vex value	overrides the clock offset rate value from the vex file
clockEpoch	date		vex value	overrides the epoch of the clock rate value; must be present if clockRate or clockOffset parameter is set
deltaClock	float	us	0.0	adds to the clock offset (either the vex value or the clockOffset above
deltaClockRate	float	us/s	0.0	adds to the clock rate (either the vex value or the clockRate above
X	float	m	vex value	change the X coordinate of the antenna location
Y	float	m	vex value	change the Y coordinate of the antenna location
Z	float	m	vex value	change the Z coordinate of the antenna location
format	string			force format to be one of VLBA, MKIV, Mark5B, S2, or one of the VDIF types
file	[string]		(none)	a comma separated list of data files to correlate
filelist	string			a filename listing files for the DATA TABLE
networkPort	int			The eVLBI network port to use for TCP/UDP. A non-number indicates raw mode attached to an ethernet interface. Both force NETWORK media type in `.input` file.
windowSize	int			TCP window size in kilobytes for eVLBI. Set to < 0 in bytes for UDP
UDP_hMTU	int			Same as setting windowSize to negative of value. For raw mode, the number of bytes to strip from ethernet frame.
vsn	string			override the Mark5 Module to be used
zoom	string			uses the global zoom configuration with matching name for this antenna; `zoom=Zoom1` will match ZOOM block called `Zoom1`
addZoomFreq	string			adds a zoom band with specified freq/bw as shown: freq1810.0/bw4.0[/specAvg4][/noparentfalse]
freqClockOffs	[float]	microsec		adds clock offsets to each recorded frequency using the format: `freqClockOffs=`*f1,f2,f3,f4*; must be same length as number of recorded freqs, first value must be zero
loOffsets	[float]	Hz		adds LO offsets to each recorded frequency using the format: `loOffsets=`*f1,f2,f3,f4*; must be same length as number of recorded freqs.
tcalFreq	int	Hz	0	enables switched power detection at specified frequency
phaseCalInt	int	MHz	1	zero turns off phase cal extraction, positive value is the interval between tones to be extracted
toneGuard	float	MHz	0.125 of bw	when using toneSelection *smart* or *most* don't select tones within this range of band edge, if possible
toneSelection	string		`smart`	tone selection algorithm; see below
sampling	string		`REAL`	set to `COMPLEX` for complex sampled data
fake	bool		False	enable a fake data source

Possible values of "tone Selection" are:

smart	write the 2 most extreme tones at least toneGuard from band edge (default)
vex	write the tones listed in the vex file to FITS
none	don't write any tones to FITS
all	write all extracted tones to FITS
ends	write the 2 most extreme tones to FITS
most	write all tones not closer than toneGuard to band edge

Setup sections are enclosed in braces after the word SETUP and a name given to this setup section. The setup name is referenced by a RULE section (see below). A setup with the special name `default` will be applied to any scans not otherwise assigned to setups by rule sections. If no setup sections are defined, a setup called `default`, with all default parameters, will be implicitly created and applied to all scans. The order of setup sections is immaterial.

Name	Type	Units	Defaults	Comments
tInt	float	sec	2	integration time
FFTSpecRes	float	MHz	0.125	frequency resolution of FFT
specRes	float	MHz	0.5	output freq res (must be mult. of FFTSpecRes
nChan	int		16	number of channels per spectral window; must be $5^m \cdot 2^n$
specAvg	int		1	how many channels to average together after correlation
fringeRotOrder	int		1	the fringe rotation order: 0=post-F, 1=linear, 2=quadratic
strideLength	int		16	number of channels to “stride” for fringe rotation, etc.
xmacLength	int		128	number of channels to “stride” for cross-multiply accumulations
numBufferedFFTs	int		1	number of FFTs to do in a row for each datastream, before XMAC
doPolar	bool		True	correlate cross hands when possible
postFFringe	bool		False	do fringe rotation after FFT?
binConfig	string		none	if specified, apply this pulsar bin config file to this setup
freqId	[int]		[] = all	frequency bands to correlate

Note that either “FFTSpecRes” and “specRes” can be used, or “nChan” and “specAvg” can be used, but the two sets cannot be mixed.

Zoom channels can be configured in a special section and referenced from ANTENNA sections to minimize complexity of the `.v2d` file. Each ZOOM section has a name and one or more “addZoomFreq” parameters, with the same format as they would have in the ANTENNA block.

Earth Orientation Parameter (EOP) data can be provided via one or more EOP sections. EOP data should be provided either in the `.v2d` file or in the vex file, but not both. Normally the vex file would be used to set EOP values, but there may be cases (eVLBI?) that want to use the vex file from `sched` without any modification. Like ANTENNA and SOURCE sections, each EOP section has a name. The name must be in a form that can be converted directly to a date (see above for legal date formats). Conventional use suggests that these dates should correspond to 0 hours UT; deviation from this practice is at the users risk. There are four parameters that should all be set within an EOP section:

Name	Type	Units	Defaults	Comments
tai_utc	float	sec		TAI minus UTC; the leap-second count
ut1_utc	float	sec		UT1 minus UTC; Earth rotation phase
xPole	float	arcsec		X component of spin axis offset
yPole	float	arcsec		Y component of spin axis offset

A rule section is used to assign a setup to a particular source name, calibration code (currently not supported), scan name, or vex mode. The order of rule sections *does* matter as the order determines the priority of the rules. The first rule that matches a scan is applied to that scan. The correlator setup used for scans that match a rule is determined by the parameter called “setup”. A special setup name `SKIP` causes matching scans not to be correlated. Any parameters not specified are interpreted as fully inclusive. Note that multiple rule sections can reference the same setup section. Multiple values may be applied to any of the parameters except for “setup”. This is accomplished by comma separation of the values in a single assignment or with repeated assignments. Thus

```
RULE rule1
{
  source = 3C84,3C273
  setup = BrightSourceSetup
}
```

is equivalent to

```
RULE rule2
{
  source = 3C84 3C273
  setup = BrightSourceSetup
}
```

is equivalent to

```
RULE rule3
{
  source = 3C84
  source = 3C273
  setup = BrightSourceSetup
}
```

The names given to rules (e.g., rule1, rule2 and rule3 above) are not used anywhere (yet) but are required to be unique.

Name	Type	Units	Comments
scan	[string]		one or more scan name, as specified in the vex file, to select with this rule
source	[string]		one or more source name, as specified in the vex file, to select with this rule
calCode	[char]		one or more calibration code to select with this rule
mode	[string]		one or more modes as defined in the vex file to select with this rule
setup	string		The name of the SETUP section to use, or SKIP if this rule describes scans not to correlate

Note that source names and calibration codes reassigned by source sections are not used. Only the names and calibration codes in the vex file are compared.

There are currently two modes of operation supported by `vex2difx`. The mode used in the vast majority of situations is called `normal` and is the default if none is specified. Currently one alternative mode, `profile`, is supported. This mode is useful for generating pulse profiles that would be useful for pulsar gating, scrunching, and binning. The difference compared to normal mode is that the standard autocorrelations are turned off and instead are computed as if they are cross correlations. This allows multiple pulsar bins to be stored. No formal cross correlations are performed. To be useful, one must create and specify a `.binconfig` file and select only the pulsar(s) from the experiment.

Philips (2022) provides more complete information and examples.

1.5 yyyyyy_N.input

This section describes the `.input` file format used by `mpifxcorr` to drive correlation. Because NRAO-DiFX 1.0 uses a non-standard branch of `mpifxcorr` some of the data fields will differ from those used in the official version, either in parameter name or in the available range of values. Currently the parameters must be in the order listed here. To get the most out of this section it is advisable to look at an actual file while reading. An example file is stashed at `http://www.aoc.nrao.edu/~wbrisken/NRAO-DiFX-1.1/` . In the tables below, numbers are assumed to floating point unless otherwise stated.

Note that the input file format has undergone a few minor changes since NRAO-DiFX version 1.0.

1.5.1 Common settings table

Below are the keywords and allowed values for entries in the common settings table. This table begins with header

```
# COMMON SETTINGS ##!
```

This is always the first table in a `.input` file.

Key	Units or allowed values	Comments
CALC FILENAME	string	name and full path to `.calc` file
CORE CONF FILENAME	string	name and full path to `.threads` file
EXECUTE TIME (SEC)	integer seconds	observe time covered by this `.input` file
START MJD	integer MJD	start date
START SECONDS	integer seconds	start time
ACTIVE DATASTREAMS	integer ≥ 2	number of antennas (*nAntenna*)
ACTIVE BASELINES	integer ≥ 1	number of baselines to correlate (*nBaseline*)
VIS BUFFER LENGTH	integer ≥ 1	the number of concurrent integrations to allow
OUTPUT FORMAT	boolean	always `SWIN` here
OUTPUT FILENAME	string	name of output `.difx` directory

Typically, $nBaseline = nAntenna \cdot (nAntenna - 1)/2$. Autocorrelations are not included in this count.

1.5.2 Configurations table

Below are the keywords and allowed values for entries in the configurations table. This table begins with header

```
# CONFIGURATIONS ###!
```

Two indexes are used for repeated keys. The index over datastream (antenna) is *d*, running from 0 to *nAntenna* - 1 and the index over baseline is *b*, running from 0 to *nBaseline* - 1.

Key	Units or allowed values	Comments
NUM CONFIGURATIONS	integer ≥ 1	number of modes in file (*nConfig*)
CONFIG NAME	string	name of configuration
INT TIME (SEC)	seconds	integration time
SUBINT NANOSECONDS	nanosec	amount of time to process as one subintegration
GUARD NANOSECONDS	nanosec ≥ 0	amount of extra data to send for overlap
FRINGE ROTN ORDER	int	0 is post-FFT, 1 is delay/rate, ...
ARRAY STRIDE LENGTH	int	used for optimized fringe rotation calculations
XMAC STRIDE LENGTH	int	number of channels to cross multiply in one batch (must evenly divide into number of channels)
NUM BUFFERED FFTS	int	number of FFTs to cross-multiply in one batch
WRITE AUTOCORRS	boolean	enable auto-correlations; *TRUE* here
PULSAR BINNING	boolean	enable pulsar mode
PULSAR CONFIG FILE	string	(*only if BINNING is True*)
PHASED ARRAY	boolean	set to FALSE (placeholder for now)
DATASTREAM *d* INDEX	integer ≥ 0	DATASTREAM table index, starting at 0
BASELINE *b* INDEX	integer ≥ 0	BASELINE table index, starting at 0

1.5.3 Rule table

The rule tables describes which configuration will be applied at any given time. Usually this filters on scan attributes such as source, but can also be done in a time-based manner (start and stop times). A time for which no configuration matches will not be correlated. If more than one rule matches a given time, they must all refer to the same configuration.

This table begins with header

```
# RULES ############!
```

The table below uses *r* to represent the rule index, which ranges from 0 to *nRule* - 1. Optional keys are identified with a $\star$.

	Key	Units or allowed values	Comments
	RULE r CONFIG NAME	string	name to associate with this rule
⋆	SOURCE	string	source to match
⋆	SCAN ID	string	scan name to match
⋆	CALCODE	string	cal code to match
⋆	QUAL	string	source qualifier to match
⋆	MJD START	string	earliest time to match
⋆	MJD STOP	string	latest time to match

1.5.4 Frequency table

Below are the keywords and allowed values for entries in the frequency table which defines all possible sub-bands used by the configurations in this file. Each sub-band of each configuration is mapped to one of these through a value in the datastream table. Each entry in this table has three parameters which are replicated for each frequency table entry. This table begins with header

```
# FREQ TABLE #######!
```

The table below uses f to represent the frequency index, which ranges from 0 to *nFreq* - 1 and t to represent pulse cal tone index, which ranges from 0 to $nTone_f$.

Key	Units or allowed values	Comments
FREQ ENTRIES	integer ≥ 1	number of frequency setups (*nFreq*)
FREQ (MHZ) f	MHz	sky frequency at band edge
BW (MHZ) f	MHz	bandwidth of sub-band
SIDEBAND f	`U` or `L`	net sideband of sub-band
NUM CHANNELS f	integer ≥ 1	initial number of channels (FFT size, $nFFT$, is twice this)
CHANS TO AVG f	integer ≥ 1	average this many channels before generating output spectra)
OVERSAMPLE FAC. f	integer ≥ 1	total oversampling factor of baseband data
DECIMATION FAC. f	integer ≥ 1	portion of oversampling to handle by decimation
PHASE CALS f OUT	integer ≥ 0	number of phase cals to produce ($nTone_f$)
		The row below is duplicated $nTone_f$ times.
PHASE CAL f/t INDEX	integer	tone number of band

1.5.5 Telescope table

Below are the keywords and allowed values for entries in the telescope table which tabulates antenna names and their associated peculiar clock offsets, and the time derivatives of these offsets. Much of the other antenna-specific information is stored in the datastream table. Each datastream of each configuration is mapped to one of these through a value in the datastream table. Each entry in this table has three parameters which are replicated for each telescope table entry. This table begins with header

```
# TELESCOPE TABLE ##!
```

The table below uses a to represent the antenna index, which ranges from 0 to *nAntenna* - 1 and c to represent clock coefficient, ranging from 0 to $nCoeff_a$.

Key	Units or allowed values	Comments
TELESCOPE ENTRIES	integer ≥ 1	number of antennas (*nAntenna*)
TELESCOPE NAME a	string	abbreviation of antenna name
CLOCK REF MJD a	double	date around which the following polynomial is expanded
CLOCK POLY ORDER a	int ≥ 0	polynomial order of telescope clock model ($nCoeff_a$
CLOCK COEFF a/c	μsec/sec^c	clock model polynomial coefficient

1.5.6 Datastream table

The datastream table begins with header

```
# DATASTREAM TABLE #!
```

The table below uses f to represent recorded frequencies, which ranges from 0 to *nFreq* - 1. A second index, z, is used to iterate over zoom bands, ranging from 0 to *nFreq* - 1. A third index, i, is used to cover the range 0 to *nBB* - 1, where the total number of basebands is given by $nBB \equiv \sum_f nPol_f$. In the DiFX system, all sub-bands must have the same polarization structure, so $nBB = nFreq \cdot nPol$. This index is reused for the zoom bands in an analogous manner.

Key	Units or allowed values	Comments
DATASTREAM ENTRIES	integer ≥ 1	number of antennas (*nDatastream*)
DATA BUFFER FACTOR	integer ≥ 1	
NUM DATA SEGMENTS	integer ≥ 1	
TELESCOPE INDEX	integer ≥ 0	telescope table index of datastream
TSYS	Kelvin	if zero (normal in NRAO usage), don't scale data by *tsys*
DATA FORMAT	string	data format
QUANTISATION BITS	integer ≥ 1	bits per sample
DATA FRAME SIZE	integer ≥ 1	bytes in one frame(or file) of data
DATA SAMPLING	string	`REAL` or `COMPLEX`
DATA SOURCE	string	`FILE`, `MODULE` for Mark5 playback
		`FAKE` for benchmarking mode
FILTERBANK USED	boolean	currently only `FALSE`
PHASE CAL INT (MHZ)	int	pulse cal comb frequency spacing, or 0 if no pulse cal tones
NUM RECORDED FREQS	integer ≥ 0	number of different frequencies recorded for this datastream
REC FREQ INDEX f	integer ≥ 0	index to frequency table
CLK OFFSET f (us)	μsec	frequency-dependent clock offset
FREQ OFFSET f (us)	μsec	frequency-dependent LO offset
NUM REC POLS f	1 or 2	for this recorded frequency, the number of polarizations
REC BAND i POL	R or L	polarization identity
REC BAND i INDEX	integer ≥ 1	index to frequency setting array above; *nBB* per entry
NUM ZOOM FREQS	integer ≥ 0	number of different zoom bands set for this datastream
ZOOM FREQ INDEX z	integer ≥ 0	index to frequency table
NUM ZOOM POLS z	1 or 2	for this recorded frequency, the number of polarizations
ZOOM BAND i POL	R or L	polarization identity
ZOOM BAND i INDEX	integer ≥ 1	index to frequency setting array above; *nBB* per entry

1.5.7 Baseline table

In order to retain the highest level of configurability, each baseline can be independently configured at some level. This datastream table begins with header

```
# BASELINE TABLE ###!
```

The baseline table has multiple entries, each one corresponding to a pair of antennas, labeled `A` and `B` in the table. For each of *nBaseline* baseline entries, *nFreq* sub-bands are processed, and for each a total of *nProd* polarization products are formed. Indexes for each of these dimensions are b, f and p respectively, each starting count at 0. Within the DiFX context, all baselines must have the same *nFreq* and *nProd*, though this is not a requirement of `mpifxcorr` in general.

Key	Units or allowed values	Comments
BASELINE ENTRIES	integer ≥ 1	number of entries in table, *nBaseline*
D/STREAM A INDEX b	integer ≥ 0	datastream table index of first antenna
D/STREAM B INDEX b	integer ≥ 0	datastream table index of second antenna
NUM FREQS b	integer ≥ 1	number of frequencies on this baseline, $nFreq_b$
POL PRODUCTS b/f	integer ≥ 1	number of polarization products, $nProd_b$
D/STREAM A BAND p	integer ≥ 0	index to frequency array in datastream table
D/STREAM B BAND p	integer ≥ 0	same as abovem, but for antenna `B`, not `A`

1.5.8 Data Table

In the following table, d is the datastream index, ranging from 0 to *nDatastream* - 1 and f is the file index ranging from 0 to $nFile_d$.

Key	Units or allowed values	Comments
D/STREAM d FILES	integer ≥ 1	number of files $nFile_d$ associated with datastream d
FILE d/f	string	name of file or module associated with datastream d

For datastreams reading off Mark5 modules, *nFile* will always be 1 and the filename is the *VSN* of the module being read.

1.6 yyyyyy_N.difx/DIFX_

The SWIN format visibilities produced by `mpifxcorr` are written to a directory with extension `.difx`. Three kinds of files can be placed in this directory as described below.

Note that the formats and naming conventions of these files is not guaranteed to stay unchanged from version to version of DiFX, and hence it is not recommended to rely on these files for archival purposes.

1.6.1 Visibility files

The bulk of the output from `mpifxcorr` is usually in the form of a binary visibility file. Usually there will be a single visibility file in this directory, but there are three ways in which multiple files may be produced: 1. a restart of the correlation, 2. if there are multiple phase centers, and 3. if there are multiple pulsar bins.

The visibility files are systematically named in the form: `DIFX_`*day*`_`*sec*`.s`*src*`.b`*bin*, where *day* is the 5 digit integer MJD of the start of visibilities, *sec* is a zero-padded 6 digit number of seconds since the MJD midnight, *src* is a 4 digit zero-padded integer specifying the phase center number (starting at 0), and *bin* is a 4 digit zero-padded integer specifying the pulsar bin number (starting at 0).

These files contain visibility data records. Each record contains the visibility spectrum for one polarization of one baseband channel of one baseline for one integration time. Each starts with a binary header and is followed by binary data.

The format of the header is shown in the table below.

Key	data type	units	comments
baseline number	int		$= (a_1 + 1) * 256 + (a_2 + 1)$ for $a_1, a_2 \geq 1$
day number	int	MJD	date of visibility centroid
seconds	double	sec	vis. centroid seconds since beginning of MJD
config index	int	≥ 0	index to `.input` file configuration table
source index	int	≥ 0	index to `.calc` file scan number
freq index	int	≥ 0	index to `.input` frequency table
antenna 1 polarization	char	`R`, `L`, `X`, `Y`	
antenna 2 polarization	char	`R`, `L`, `X`, `Y`	
pulsar bin number	int	≥ 0	
visibility weight	double	≥ 0.0	data weight for spectrum; typically ~ 1
u	double	meter	u component of baseline vector
v	double	meter	v component of baseline vector
w	double	meter	w component of baseline vector

Note that for both the header and the data, the endianness is native to the machine running `mpifxcorr`, and there are currently no provisions for processing such files on a machine of different endianness.

Following the end-of-line mark for the last header row begins binary data in the form of (real, imaginary) pairs of 32-bit floating point numbers. The `.input` file parameter `NUM CHANNELS` indicates the number of complex values to expect. In the case of upper sideband data, the first reported channel is the "zero frequency" channel, that is its sky frequency is equal to the value in the frequency table for this spectrum. The Nyquist channel is not retained. For lower sideband data, the last channel is the "zero frequency" channel. That is, in all cases, the spectrum is in order of increasing frequency and the Nyquist channel is excised.

1.6.2 yyyyyy_N.difx/PCAL_

Pulse calibration data can be extracted by `mpifxcorr`. Extraction is configured on a per-antenna basis. Data for each antenna is written to a separate file; if correlation is restarted, an additional pulse cal data file will be written.

The pulse cal data files are systematically named in the form: `PCAL_`*day_sec_ant*, where *day* is the 5 digit integer MJD of the start of visibilities, *sec* is a zero-padded 6 digit number of seconds since the MJD midnight, and *ant* is the 1 or 2 letter antenna name in capital letters. There is potential for these text files to have very long lines (more than 10,000 bytes) when many pulse cal tones are extracted.

For DiFX versions 2.3 and earlier the data format was exactly the same. This old version will be considered "version 0".

The data format being used now is similar in spirit but more convenient for `mpfixcorr` to produce and for `difx2fits` and `difx2mark4` to digest leading to broader support (in theory complete) of the various polarization, frequency, and sideband combinations allowed by DiFX. The data format is as follows:

Comment lines begin with an octothorpe (#). The first few lines of comments may contain machine-readable information in the following format:

```
# DiFX-derived pulse cal data
# File version = 1
# Start MJD =
# Start seconds =
# Telescope name =
```

Data lines always contain 6 fixed-size fields:

1. *antId* : Station name abbreviation, e.g., `LA`
2. *day* : Time centroid of measurement (MJD, including fractional portion)
3. *dur* : Duration of measurement (days)
4. *datastreamId* : The datastream index of for this data.
5. *nRecBand* : Number of recorded baseband channels
6. *nTone* : (Maximum) number of pulse cal tones detected per band per polarization

Following these fields is a variable-length arrays of numbers. This array contains the pulse cal data and consists of *nRecBand*nTone* groups of four numbers. The groups are arranged in ascending record band index (slow index) and ascending tone number (fast index) where the tone number increases away from the reference frequency; not sure what happens with dual-sideband complex! The first member of this group is the tone frequency (MHz), or -1 to indicate there was not a measurment. The second member of this group is the polarization, one of `R`, `L`, `X` or `Y`. The third and fourth are respectively the real and imaginary parts of the tone measured at the given sky frequency.

Text after a comment character (#) are ignored.

1.7 yyyyyy_N.im

The `.im` file contains polynomial models used by `difx2fits` in the creation of `FITS` files. After a header that is similar to that of a `.rate` file, the contents are organized hierarchically with scan number, sub-scan interval, and antenna number being successively faster-incrementing values. The keys and allowed values in this section are summarized below: Note that the values of the delay polynomials in this file have the opposite sign as compared to those generated by CALC and those stored in `.FITS` files. Keys preceded by ⋆ are optional. Note that all polynomials are expanded about their `MJD, SEC` start time and use seconds as the unit of time.

Key	Units or allowed values	Comments
⋆ CALC SERVER	string	name of the calc server computer used
⋆ CALC PROGRAM	integer	RPC program ID of the calc server used
⋆ CALC VERSION	integer	RPC version ID of the calc server used
START YEAR	integer	calendar year of START MJD
START MONTH	integer	calendar month of START MJD
START DAY	integer	day of calendar month of START MJD
START HOUR	integer	hour of START MJD
START MINUTE	integer	minute of START MJD
START SECOND	integer	second of START MJD
POLYNOMIAL ORDER	2, 3, 4 or 5	polynomial order of interferometer model *order*
INTERVAL (SECS)	integer	interval between new polynomial models
ABERRATION CORR	{ `UNCORRECTED` / `APPROXIMATE` / `EXACT`	level of u, v, w aberration correction
NUM TELESCOPES	integer ≥ 1	number of telescopes ($nTelescope$) The row below is duplicated $nTelescope$ times.
TELESCOPE t NAME	string	upper case antenna name abbreviation
NUM SCANS	integer ≥ 1	number of scans ($nScan$). Everything below is duplicated $nScan$ times.
SCAN s POINTING SRC	string	name of source used as pointing center
SCAN s NUM PHS CTRS ≥ 1	number of phase centers this scan (nPC_s)	
		Everything below is duplicated nPC_s times.
SCAN s PHS CTR p SRC	string	name of source defining this phase center
SCAN s NUM POLY	≥ 1	number of polynomials covering scan ($nPoly_{s,p}$) Everything below is duplicated $nPoly_{s,p}$ times.
SCAN s POLY p MJD	integer ≥ 0	the start MJD of this polynomial
SCAN s POLY p SEC	integer ≥ 0	the start sec of this polynomial
		Everything below is duplicated $nTelescope$ times.
ANT a DELAY (us)	*order*+1 numbers	terms of delay polynomial
ANT a DRY (us)	*order*+1 numbers	terms of dry atmosphere
ANT a WET (us)	*order*+1 numbers	terms of wet atmosphere
⋆ ANT a AZ	*order*+1 numbers	azimuth polynomial (deg)
⋆ ANT a EL GEOM	*order*+1 numbers	geometric (encoder) elevation (deg)
⋆ ANT a EL CORR	*order*+1 numbers	refraction corrected elevation (deg)
⋆ ANT a PAR ANGLE	*order*+1 numbers	parallactic angle (deg)
ANT a U (m)	*order*+1 numbers	terms of baseline u
ANT a V (m)	*order*+1 numbers	terms of baseline v
ANT a W (m)	*order*+1 numbers	terms of baseline w

1.8 yyyyyy_N.calc

The main use of the `.calc` file is to drive the geometric model calculations but this file also serves as a convenient place to store information that is contained in the `.fx` file but not in the `.input file` and is needed for `.FITS` file creation. In the DiFX system, one `.calc` file is created by `vex2difx` program for each `.input` file. This file is read by `calcif2`) program to produce a tabulated delay model, u, v, w values, and estimates of atmospheric delay contributions.

In brief, the parameters in this file that are relevant for correlation include time, locations and geometries of antennas, pointing of antennas (and hence delay centers) as a function of time and the Earth orientation parameters relevant for the correlator job in question. Additional parameters that are stuffed into this file include spectral averaging, project name, and information about sources such as calibration code and qualifiers. In the NRAO application of DiFX, source names are faked in the actual `.input` file in order to allow multiple different configurations for the same source. A parameter called *realname* accompanies each source name in the `.calc` file to correctly populate the source file in `.FITS` file creation.

The syntax of this file is similar to that of the `.input` file. The file consists entirely of key-value pairs separated by a colon. The value column is not constrained to start in column 21 as it is for the files used by `mpifxcorr`. There are five sections in the `.calc` file; these sections are not separated by any explicit mark in the file.

The first section contains values that are fixed for the entire experiment and at all antennas; all data in this section is scalar. In the following table, all numbers are assumed to be floating point unless further restricted. The keys and allowed values in this section are summarized below. Optional keys are identified with a ⋆. Deprecated keys that will likely be removed in an upcoming version are identified with an ×.

Key	Units or allowed values	Comments
JOB ID	integer ≥ 1	taken from `.fx` file
⋆ JOB START TIME	MJD + fraction	start time of original `.fx` file
⋆ JOB STOP TIME	MJD + fraction	end time of original `.fx` file
⋆ DUTY CYCLE	float ≤ 1	fraction of the job contained within scans
OBSCODE	string	observation code assigned to project
⋆ SESSION	short string	session suffix to OBSCODE, e.g., `A` or `BE`
⋆ DIFX VERSION	string	version of correlator, e.g. `DIFX-1.5`
⋆ DIFX LABEL	string	name of correlator install, e.g. `DIFX-WALTER`
VEX FILE	string	dir/filename of vex file used to create the job
START MJD	MJD + fraction	start time of this subjob
START YEAR	integer	calendar year of START MJD
START MONTH	integer	calendar month of START MJD
START DAY	integer	day of calendar month of START MJD
START HOUR	integer	hour of START MJD
START MINUTE	integer	minute of START MJD
START SECOND	integer	second of START MJD
⋆ SPECTRAL AVG	integer ≥ 1	number of channels to average in FITS creation
⋆ START CHANNEL	integer ≥ 0	start channel number (before averaging)
⋆ OUTPUT CHANNELS	integer ≥ 1	total number of channels to write to FITS
	$> 0.0, < 1.0$	fraction of total channels to write to FITS
⋆ TAPER FUNCTION	string	currently only `UNIFORM` is supported

The second section contains antenna(telescope) specific information. After an initial parameter defining the number of telescopes, there are *nTelescope* sections (one for each antenna), each with the following six parameters. Lowercase *t* in the table below is used to indicate the telescope index, an integer ranging from 0 to *nTelescope* - 1. Note that in cases where units are provided under the Key column, these units are actually part of the key.

Key	Units or allowed values	Comments
NUM TELESCOPES	integer ≥ 1	number of telescopes (*nTelescope*). The rows below are duplicated *nTelescope* times.
TELESCOPE t NAME	string	upper case antenna name abbreviation
TELESCOPE t MOUNT	string	the mount type: altz, equa, xyew, or xyns
TELESCOPE t OFFSET (m)	meters	axis offset in meters
TELESCOPE t X (m)	meters	X geocentric coordinate of antenna at date
TELESCOPE t Y (m)	meters	Y geocentric coordinate of antenna at date
TELESCOPE t Z (m)	meters	Z geocentric coordinate of antenna at date
⋆ TELESCOPE t SHELF	string	shelf location of module to correlate

The third section contains a table of sources. Sources are indexed from the following section describing the scans.

Key	Units or allowed values	Comments
NUM SOURCES	integer ≥ 1	number of sources (*nSource*). The rows below are duplicated *nSource* times.
SOURCE s NAME	string	name of source (possibly renamed from `.vex`
SOURCE s RA	radians	J2000 right ascension
SOURCE s DEC	radians	J2000 declination
SOURCE s CALCODE	string	usually upper case letters or blank
SOURCE s QUAL	integet ≥ 0	source qualifier

The fourth section contains scan specific information. Except for one initial line specifying the number of scans, *nScan*, this section is composed of nine parameters per scan. Each parameter is indexed by s which ranges from 0 to *nScan* - 1.

Key	Units or allowed values	Comments
NUM SCANS	integer ≥ 1	number of scans (*nScan*). The rows below are duplicated *nScan* times.
SCAN s IDENTIFIER	string	name of the scan (not of the source)
SCAN s START (S)	seconds	scan start time, relative to job start time
SCAN s DUR (S)	seconds	duration of scan
SCAN s OBS MODE NAME	string	reference to `.input` file configuration
SCAN s UVSHIFT INTERVAL (NS)	time to integrate before doing uv shifts (used mainly for multi phase-center observing)	
SCAN s AC AVG INTERVAL (NS)	averaging interval for export of fast-dump spectra (used for VFASTR)	
SCAN s POINTING SRC	integer ≥ 1 pointing center of scan	source table index identifying
SCAN s NUM PHS CTRS	integer ≥ 1	number of phase centers (*nPC*). The rows below are duplicated *nPC* times.
SCAN s PHS CTR p	integer ≥ 1	index to the source table

The fifth section contains Earth orientation parameters (EOP). Except for one initial line specifying the number of days of EOPs, *nEOP*, this section is composed of five parameters per day of sampled EOP values. Each parameter is indexed by e which ranges from 0 to *nEOP* - 1.

Key	Units or allowed values	Comments
NUM EOP	integer ≥ 1	number of tabulated EOP values (*nEOP*). The rows below are duplicated *nEOP* times.
EOP e TIME (MJD)	MJD + fraction	time of sample; fraction almost always zero
EOP e TAI_UTC (sec)	integer seconds	leap seconds accrued at time of job start
EOP e UT1_UTC (sec)	seconds	UT1 - UTC
EOP e XPOLE (arcsec)	arc seconds	X coordinate of polar offset
EOP e YPOLE (arcsec)	arc seconds	Y coordinate of polar offset

The next (completely optional) section has a table for positions and velocites of spacecraft. Each spacecraft is indexed by s and each row thereof by r.

Key	Units or allowed values	Comments
⋆ NUM SPACECRAFT	integer ≥ 0	number of spacecraft (*nSpacecraft*) Everything below is duplicated *nSpacecraft* times.
SPACECRAFT s NAME	string	name of spacecraft
SPACECRAFT s ROWS	integer ≥ 1	number of data rows, $nRow_s$ for spacecraft s The row below is repeated $nRow_s$ times.
SPACECRAFT s ROW r	7 numbers	tabulated data; see below

Each data vector of data consists of seven double precision values: time (mjd), x, y, and z (meters), and $\dot{x}$, $\dot{y}$, and $\dot{z}$ (meters per second). These values should be separated by spaces.

The final section identifies the files to be produced.

Key	Units or allowed values	Comments
IM FILENAME	string	dir/filename of `.im` file to create
FLAG FILENAME	string	dir/filename of `.flag` file to create

1.9 yyyyyy_N.difxlog

The `difxlog` program captures `DifxAlertMessage` and `DifxStatusMessage` message types that are sent from an ongoing software correlation process and writes the information contained within to a human readable text file. One line of text is produced for each received message. The first five columns contain the date and time in *ddd MMM dd hh:mm:ss yyyy* format (e.g., `Wed Apr 22 12:48:41 2009`). The sixth column contains a word describing the contents of the remainder of the line: Options are:

`STATUS` : The status of the process is described

`WEIGHTS` : The playback weights for each antenna are listed

other : This word represents an alert severity level (one of `FATAL`, `SEVERE`, `ERROR`, `WARNING`, `INFO`, `VERBOSE` and `DEBUG`) and is followed by the alert message itself.

1.10 yyyyyy_N.filelist

When using the `filelist` parameter in an `ANTENNA` section of a `.v2d`, the list of data files to correlate are stored in a text file. This is a text file containing data lines and optionally comments. Any text after the comment character (#) is ignored. A data line consists of a filename (must have complete path as can be used to find the file on the datastream node for this antenna) and optionally a start time and stop time. Start and stop times can be expressed in any of the formats supported by `vex2difx`.

1.11 yyyyyy_N.flag

The program `vex2difx` may write a `.flag` file for each `.input` file it creates. This file is referenced from the `.calc` file. This flag file is used by `difx2fits` to exclude nonsense baselines that might have been correlated. Data from nonsense baselines can occur in DiFX output when multiple subarrays are coming and going. The flag file instructs `difx2fits` to drop these data during conversion to FITS-IDI.

The format of this text file is as follows. The first line contains an integer, n, which is the number of flag lines that follow. The next n lines each have three numbers: MJD_1, MJD_2 and ant. The first two floating point numbers determine the time range of the flag in Modified Julian Days. The last integer number is the antenna number to flag, a zero-based index corresponding to the `TELESCOPE` table of the corresponding `.input` file.

1.12 yyyyyy_N.joblist

A single `.joblist` file is written by `vex2difx` as it produces the DiFX `.input` (and other) files for a given correlator pass. This file contains the list of jobs to run and some versioning information that allows improved accountability of the software versions being used. This file us used by `difxqueue` and `makefits` to ensure that a complete set of jobs is accounted for. The file is composed of two parts: a header line and one line for each job. The header line consists of a series of *key=value* pairs. Each *key* and *value* must have no whitespace and no whitespace should separate these words from their connecting `=` sign. While any number of key-value pairs may be specified, the following ones (which are case sensitive) are expected to be present:

1. `exper` : the name of the experiment, including the segment code
2. `v2d` : the `vex2difx` input file used to produce the jobs of this pass
3. `pass` : the name of the correlator pass
4. `mjd` : the modified Julian day when `vex2difx` created this file
5. `DiFX` : the version name for the DiFX deployment (the value of `$DIFX_VERSION` when `vex2difx` was run)
6. `vex2difx` : the version of `vex2difx` that was run

Each additional line contains information for one job in the pass. The columns are:

1. *jobName* : the name/prefix of the job
2. *mjdStart* : the observe start time of the job
3. *mjdStop* : the observe stop time of the job
4. *nAnt* : the number of antennas in the job
5. *maxPulsarBin* : the maximum number of pulsar bins to come from any scan in this job (usually zero)
6. *nPhaseCenters* : the maximum number of phase centers to come from any scan in this job (usually one)
7. *tOps* : estimated number of trillion floating point operations required to run the job
8. *outSize* : estimated FITS file output size (MB)

Usually the comment character `#` followed by a list of station codes is appended to the end of each line.

1.13 yyyyyy_N.machines

The `.machines` file is used by `mpirun` to determine which machines will run `mpifxcorr`. This is a text file containing a list of computers, one to a line possibly with additional options listed, on which to spawn the software correlator process. As a general rule the MPI rank, a unique number for each process that starts at 0, are allocated in the order that the computer names are listed. This general rule can break down in cases where the same computer name is listed more than once; the behavior in this case depends on the MPI implementation being used. MPI rank 0 will always be the manager process. Ranks 1 through *nDatastream* will each be a datastream process. Additional processes will be computing (core) processes. If more processes are specified for `mpirun` with the `-np` option than there are lines in this file, the file will be read again from the top, so the processes will be assigned in a cyclic fashion (again, this depends somewhat on the MPI implementation and the other parameters passed to `mpirun`; for DiFX with OpenMPI, this assumes `--bynode` is used). If the program `startdifx` is used to start the correlation process, the number of processes to start is determined by the number of lines in this file. If wrapping to the top of this file is desired, dummy comment lines (beginning with #) can be put at the end of the

`.machines` file to artificially raise the number of processes to spawn. Within DiFX, this file is typically produced by `genmachines`. Keep in mind that this file is directly read by the MPI execution program `mpirun` and the format of the file may differ depending on the MPI implementation that you are using. With OpenMPI appending `slots=1 max-slots=1` to the end of each line ensures that a single instance of `mpifxcorr` is run on that machine. If both a datastream process and a core process are to be run on the same computer, then using options `slots=1 max-slots=2` might be appropriate.

1.14 yyyyyy_N.threads

The `.threads` file tells `mpifxcorr` how many threads to start on each processing node. Within DiFX, this file is typically produced by `genmachines`. The `.threads` file has a very simple format. The first line starts with `NUMBER OF CORES:`. Starting at column 21 is an integer that should be equal to the number of processing nodes (*nCore*) specified in the corresponding `.machines` file. Each line thereafter should contain a single integer starting at column 1. There should be *nCore* such lines.

2 Credit

Abstract and sections 1.1–1.3 are written by Leonid Petrov. Other sections are written by Walter Bisken and extracted from the DiFX User Guide accessible in subdirectory `doc/userguide/trunk` of the DiFX source code.

DiFX documentation is availble online (Philips, 2022).

3 References

References

Deller, A. T., W. F. Brisken, C. J. Phillips, J. Morgan, W. Alef, R. Cappallo, E. Middelberg, J. Romney, H. Rottmann, S. J. Tingay, and R. Wayth (2011), "DiFX-2: A More Flexible, Efficient, Robust, and Powerful Software Correlator." *Publ. Astron. Soc. Pacific*, 123, 275 doi: 10.1086/658907.

Deller, A. T., S. J. Tingay, M. Bailes, and C. West (2007), "DiFX: A Software Correlator for Very Long Baseline Interferometry Using Multiprocessor Computing Environments." *Publ. Astron. Soc. Pacific*, 119, 318–336 doi: 10.1086/513572.

Himwich, Edward (2021), "Vex definition." Technical report, NRAO, URL `https://safe.nrao.edu/wiki/bin/view/VLBA/Vex2doc`.

Philips, Christian (2022), "Difx developer pages." Technical report, CSIRO, URL `https://www.atnf.csiro.au/vlbi/dokuwiki/doku.php/difx/start`.