VLBA Sensitivity Upgrade MEMO 20

Difx Operator Interface (DOI) Specification

Walter Brisken

National Radio Astronomy Observatory

April 18, 2008

Introduction

1 Top level requirements

This section describes the top-level requirements of the DiFX Operator Interface (DOI) system for running NRAO-DiFX. The priorities for each requirement are noted within square brackets with priority 1 being the highest. Subrequirements without specified priority inherit that from the parent requirement. This is a "living document". Expect changes to this document with time.

1-R1 (1) Must run on 32 and 64 bit Intel Linux

The base system will be 64 bit Intel Linux, likely based on RedHat Enterprise Linux ver 5.1 or later. The DOI must run on such systems.

1-R2 (1) Full monitoring and control capabilities

The DOI must be able to control and monitor the running of NRAO-DiFX without need for additional command line or GUI tools. Note that this requirement does not extend to maintenance and debugging.

1-R3 (2) Multiple instances

It must be possible to run multiple instances of the DOI, possibly on different computers, at the same time.

1-R4 (1) Control mode

The standard DOI instance for an operator will be control mode, giving full control of running jobs and monitoring the processing. If not in control mode, it is said to be in monitor mode.

1-R4.1 (1) Single control mode instance
Only one instance of control mode should be allowed at any one time.

1-R4.2 (2) Control mode access control
Control mode should be password protected. Only one userid (called the DiFX user) will be allowed to run in control mode. (Note — this authentication scheme is still To Be Determined)

1-R4.3 (2) Demotion to monitor mode
It should be possible to switch to monitor mode without any extra permission.

1-R4.4 (2) Control mode indicator
Each GUI window should identify itself as being in control mode if it is. This can be done in the window title bar or via a status bar at the bottom of each window.

1-R5 (2) Monitor mode

It must be possible to run an instance of the GUI in monitor mode, a mode where full visualization of the current state of the correlator and projects under control is possible, but in which control of the correlator or components (resources) there of is not possible.

1-R5.1 Access control
Any user in a specific access group should be granted permission to run in monitor mode.

1-R5.2 Promotion to control mode
If it was the DiFX user that started the DOI and the proper password supplied, promotion to control mode shall be granted.

1-R5.3 Greyed-out options
Any buttons that cause the correlator or any if its resources to change state shall be greyed-out when in monitor mode.

1-R6 (2) Integration with OMS

The DOI must be integrated with the Observation Management System (OMS) at the same level that the hardware correlator is integrated.

1-R7 (1) Logging

A centralized logging system must permit the writing of a log.

1-R7.1 Log file cycling
A new log file should be started every UT day midnight. The names of the log files should contain the current Modified Julian Day (MJD).

1-R7.2 Timestamping
All log file entries should be accompanied by a timestamp in Universal Time. The top of each day's log file should explicitly include the date corresponding to that file, even though the date will be encoded in the filename.

1-R8 Start-up sequence

The DOI started in control mode should do the following on start-up:

1-R8.1 Log start-up
Whenever a control mode DOI is started (whether a new DOI or via promotion from monitor mode), this should be noted in the log file.

1-R8.2 Determine state
Upon startup, the current state of the software correlator is to be determined. No control operations should be allowed for an initial period long enough to ensure that no job is being correlated.

1-R8.3 Start-up while correlating
Normally the DOI will not be restarted while correlation is in progress, but the ability for this to happen without consequences is required.

1-R9 DOI Components

This remainder of this document describes various "components". These portions of the overall program do not necessarily reflect code organization but rather functional organization.

1-R9.1 (1) Components always active
Each appropriate component shall be active, regardless of whether the window(s) for that component are visible or not. The state information maintained by each component should always be up to date.

1-R9.2 (1) Multiple component instances

For some component types (such as the job monitor), multiple active instances must be supported within a single instance of the DOI.

1-R10 (1) Config file

A configuration file shall contain runtime parameters for the DOI.

1-R10.1 (1) Editability
The configuration file should be editable with a text file.

1-R10.2 (1) Extensibility
The ability to add new parameters without breaking the format is required.

2 Resource Manager

The resource manager component maintains and displays the status of the software correlator computer cluster nodes.

2-R1 (1) The resource manager must always be active, even if display of its DOI window is disabled.

2-R2 (1) States

At any given time a cluster member can be in a particular state. This state is both informative to the operator and indicates to the DOI which actions can be taken on that cluster member at a given time.

2-R2.1 Generic states

The following are possible states for any cluster member:

2-R2.1.1 IDLE

This resource is not assigned to any processing.

2-R2.1.2 ASSIGNED

The resource is assigned to at least one process.

2-R2.1.2.1 (3) Multiple assignments

It is possible that a processing unit can have its multiple CPU cores assigned to more than one job at a time. Supporting this mode is low priority.

2-R2.1.3 ERROR

An error is associated with the process. There should always be an error message associated with the ERROR state that can be accessed by the operator and the message should be sent to the log.

2-R2.1.4 OFFLINE

When the unit is disabled (see below), the state is always OFFLINE.

2-R2.2 Mark5 unit states

Direct Mark5 module playback has its own sub-states that indicate to the operator the status of the Mark5 unit. Special attention is payed to the Mark5s due to their high rate of failures due to module malfunctions and occasional driver instabilities.

2-R2.2.1 Opening

The opening of Mark5 device is being attempted.

2-R2.2.2 Open

The opening of the Mark5 device was successful. Next state will be GetDirectory.

2-R2.2.3 Close

The Mark5 device is closed.

2-R2.2.4 GetDirectory

The directory of scans on the module is being retrieved. This can take many minutes in some cases.

2-R2.2.5 GotDirectory

The directory recovery was successful. Next state will be Play.

2-R2.2.6 Play

Data is being played off the module

2-R2.2.7 Idle

Though perhaps assigned, the Mark5 unit is not doing anything

2-R2.2.8 Error

An error occurred. An error message will accomany any indication of the ERROR state which should be remembered. All error messages should be logged.

2-R3 (1) Status of cluster members shall be maintained

2-R3.1 Cluster membership

The list of cluster members should be maintained in a cluster configuration. This file shall contain the following information for each cluster member:

2-R3.1.1 Name of machine
This will typically be a short name like mark5fx20 or swc003. This field should allow for names that include fully qualified computer names, such as mark5fx07.aoc.nrao.edu.

2-R3.1.2 Number of processing cores

2-R3.1.3 Whether machine is Mark5 or not

2-R3.1.4 Whether machine is currently enabled or disabled
A disabled machine will not be used in data processing and will not get any commands from the DOI at all. The resource manager shall continue to accept messages sent from the disabled machines, but will not report errors if expected signals are not received.

2-R3.2 CPU and memory load
State information will be sent from all cluster members via multicast XML messages of type *DifxLoadMessage*. These messages will typically be sent by a program called loadmon which will always be running on cluster members.

2-R3.2.1 Message contents

2-R3.2.1.1 CPU load
The reported number is the average number of processes being scheduled on the CPUs inside the cluster member.

2-R3.2.1.2 Memory usage
The number of kilobytes of used memory and the number of kilobytes of system memory will be reported.

2-R3.2.2 Time tagging
The messages shall be time-tagged (MJD / UT) upon arrival to a precision of 1 second or better.

2-R3.2.3 Message frequency
The load message shall be sent from each cluster member at a specified interval (perhaps 10 seconds). A warning message that is visible to the operator shall be issued if a message from particular enabled machine is not received in three multiples of this interval. The state of the machine shall be set to ERROR.

2-R3.3 Mark5 status message
State information about the Mark5 units is sent either by the `mpifxcorr` correlator process itself or by a standalone program called `mk5agent`. This is sent in an XML message of type *Mark5StatusMessage*

2-R3.3.1 Mark5 status request
When a Mark5 unit is not assigned to a job for playback (including cases when the Mark5 is being used as a processing node but not a datastream node), the Mark5 state information must be requested by sending an XML request message to the `mk5agent` on the particular unit. (This is an idea still in development)

2-R3.3.2 Message contents

2-R3.3.2.1 Bank A Volume Serial Number (VSN)
The 8 character name of the Mark5 module in bank A of the Mark5 unit, or "none" if loaded.

2-R3.3.2.2 Bank B VSN

2-R3.3.2.3 Status word
A 32 bit value represented in hexadecimal coding various status bits.

2-R3.3.2.4 Active Bank
A character A, B, or blank (for neither) indicating which bank of the Mark5 unit is currently selected as active. The active bank is only of concern when playback is occuring.

2-R3.3.2.5 State
The state of the Mark5 unit as described above under "Mark5 unit states" (Req. 2-R2.2).

2-R3.3.2.6 Scan Number
The currently selected scan number. This number is only applicable during playback and refers only to the active bank.

2-R3.3.2.7 Position
The byte pointer from which playback is occurring. This value must be capable of accurately storing integer values up to 10^{15}. This number is acurate only applicable during playback; its values shall remain latched at the last reported value when playback stops

2-R3.3.2.8 Rate
The approximate playback rate, averaged over an interval (of about 1 second), measured in Mbps. The value is a floating point number and is only applicable during playback.

2-R3.3.2.9 Data MJD
The most recent requested data time, in MJD + fractional day. The value should be stored in a double precision floating point number. It is only applicable during data playback.

2-R4 (1) Access to state information

The state information shall be accessible to any other component in the DOI program.

2-R5 (1) Display of information

2-R6 (1) Manipulation of cluster configuration file

The DOI should allow the following changes to be made to the cluster configuration file. Note that changing or removing resources that are allocated to a currently running correlator job is not to be allowed.

2-R6.1 (1) Change enabled status
2-R6.2 (2) Add new machine row
2-R6.3 (2) Remove existing machine row

2-R7 (1) Actions on selected resources

2-R7.1 Reboot
This would probably be via ssh.

2-R7.2 Poweroff
This would probably be via ssh.

2-R7.3 Mark5 Reset
Done by sending a particular request to the mk5agent program running on the unit.

2-R7.3.1 Respect state of operation
Resetting the Mark5 unit should not be allowed if a job is successfully using that unit at the time of request.

3 Job Manager

The job manager maintains and displays the current state of correlation of a particular job.

3-R1 (1) Active job monitoring

A job manager must be running at all times for any job that is being correlated.

3-R2 (1) Visibility

The window should be able to be hidden. Even when the GUI for an active job is hidden (closed), the job monitoring must continue.

3-R3 (2) Multiple job managers

It should be possible to have job managers active and displaying data for more than one job at a time.

3-R4 (1) Sources of information

The job monitor should get its information from a small set of sources:

3-R4.1 The job .calc file

The .calc file that is created by job2difx contains vital information about the project. Note that the information currently contained in the .calc file is to be moved at some point in the future, but all of the information will continue to be available. Key information includes:

3-R4.1.1 Project code

The project code assigned to the experiment, such as BC1120A or GB057. In general each project will have multiple correlator jobs associated with it. The project code should be treated as a character string not containing whitespace that could be at least 8 characters long.

3-R4.1.2 Job ID

The job ID is a unique (within a project) name given to a particular job running on the correlator. Each job ID will come with its own set of files used by `mpifxcorr` and each will produce one visibility file output. The job ID should be treated as a string up to 16 characters long.

3-R4.1.3 List of stations in job

A name for each antenna in the job is included in the .calc file.

3-R4.1.4 Shelf locations

For each Mark5 Module used in the job a shelf location is included in the .calc file.

3-R4.2 The job .input file

The .input file is used to run the correlator. Some information not present in the .calc file will be needed, including:

3-R4.2.1 Module VSNs

The VSN of the Mark5 Module containing data from each station in the project is contained in the DATA table of the .input file. Projects are divided into jobs such that there can be only one module per station in a given job.

3-R4.2.2 Job start time

The job start time (refering to the wall clock time during the observation).

3-R4.2.3 Job duration

Number of seconds of observe time represented by this job.

3-R4.3 Multicast XML documents

During correlation, `mpifxcorr` will send multicast messages containing the current status of correlation.

3-R4.3.1 DifxStatusMessage

The DifxStatusMessage XML document contains the following information. Note that the receipt of such a document should be logged with a timestamp.

3-R4.3.1.1 Job State

The job state describes the state of a running `mpifxcorr` instance. In addition to informing the operator of the current condition, the current state is very important for debugging when problems arise. The project state can be one of:

3-R4.3.1.1.1 Spawning

The processes are being spawned via ssh. This message is sent by `startdifx`, not `mpifxcorr`.

3-R4.3.1.1.2 Starting

The processes have been spawned and mpifxcorr is initializing

3-R4.3.1.1.3 Running

The process is currently running.

3-R4.3.1.1.4 Ending
The correlation is about to finish, successfully.

3-R4.3.1.1.5 Done
The correlation concluded successfully. Receipt of this message is the only way to ensure that correlation was successful.

3-R4.3.1.1.6 Aborting
The correlator is stopping before it starts due to some configuration error. Note an error message should be sent to the log.

3-R4.3.1.1.7 Terminating
A termination signal (SIGINT) was received by the manager node and the correlator is shutting down gracefully, but early. The output visibility file will be incomplete. Probably the output file should be deleted.

3-R4.3.1.1.8 Terminated
Early termination was successful.

3-R4.3.1.1.9 MpiDone
The `mpifxcorr` processes have ended. This message is sent by `startdifx`, not `mpifxcorr`. Receipt of this message should cause resources allocated to this job to be freed. All jobs will finish with a message of this type being emitted, whether or not the job was successfully completed. Receipt of this document type should not change the state assigned to the job.

3-R4.4 Information from Queue Manager
In addition to the job states listed under Req. 3-R4.3.1.1, the state of a job can take on other values. If the job is in the queue and has not yet started running, it should take on one of the values listed under Req. 5-R2.1.

3-R4.5 (2) File existence
The existence of the following files should be tested to determine the readiness for correlation. Note that <jobId> should be replaced with the job ID string. See external documentation for descriptions of these files and their use.

3-R4.5.1 <jobId>`.input`
3-R4.5.2 <jobId>`.calc`
3-R4.5.3 <jobId>`.uvw`
3-R4.5.4 <jobId>`.delay`

3-R5 (1) Allowable job states

A given job can take on any of the states that are sent from an `mpifxcorr` process, listed under *DifxStatusMessage* (see Req. 3-R4.3.1) or the following states which are assigned by the GUI, possibly in response to the result of correlation:

3-R5.1 Ready
The job is ready to run, meaning that it has not yet been run, all the modules needed for correlation are loaded, and none of the loaded modules are in a Mark5 unit that is being used to playback data for another job (note this last condition applies whether or not the module is the currently active one or not).

3-R5.2 Waiting
The job is ready to run, but there is a conflict with a currently running job.

3-R5.3 Conflict
Two modules for the job are currently loaded in the same Mark5 unit and thus the job cannot be run. (Note that this restriction may be dropped sometime at the expense of slower correlation)

3-R5.4 NotReady
The job cannot be run because at least one required module is not currently loaded

3-R5.5 Killed
The GUI should give this name to a job after the MpiDone state is seen if the correlation was not successful.

3-R5.6 Complete
The job was run successfully. A job in this state cannot be added back to the queue.

3-R6 (1) Information to display

3-R6.1 Project information
3-R6.2 Project state information
3-R6.3 Derived values
3-R6.3.1 Estimated duration of job execution
This is the observe time represented by the job divided by the estimated speed-up factor
3-R6.3.2 Progress
The fraction complete (with time remaining) should be displayed.
3-R6.3.3 Disc usage
Estimated space to be used by correlation products
3-R6.4 Error messages
Error messages received for active jobs should be stored in local memory while a job manager for that job remains active. The ability to scroll up to see any error message received during the job execution.

3-R7 (1) GUI functionality

The GUI should allow the following actions to be performed

3-R7.1 Stop correlation
This will result in the clean termination of the job. A dialog window should verify intent of the operator.
3-R7.2 Start
This button will immediately start correlation of the job if it is in the Ready state. The button should be greyed out if not Ready.
3-R7.3 (2) Error reporting
It should be possible to automatically generate an email error report that includes all of the collected information about the job, the resources last assigned to the job, the wall clock time/date (UT) of job start, and all error messages received during the execution of the job.

4 Project Manager

The project manager provides project-based management of correlator processing. The project is the fundamental unit of scientific data. Note that "project" and "experiment" are often used interchangably. Formally speaking, what is refered to here as a project is a "scheduling block" in the parlance of EVLA and ALMA, or a "segment" in VLBA operations terminology.

4-R1 (2) Multiple project manager windows

It should be possible to view multiple project manager GUI windows at one time.

4-R2 (1) Maintain state of a project

4-R2.1 Project directory
All the files (both input files and correlated output) for a particular project will reside in a single, heirarchical directory. This directory will be an immediate subdirectory of a directory pointed to by environment variable JOB_ROOT.
4-R2.2 Project name
The project name will always be the name of the top level project directory. For example, if the project directory is /home/swc001/difx/bm270 (or $JOB_ROOT/bm270), then the project name would be bm270 .

4-R2.3 List of jobs

Each project will have between 1 and many job files. Each job will have .input and .calc available before correlation. The names of the jobs are these filenames without the extensions. For example a job with name job4621.00.0 will have an .input file called job4621.00.0.input and a .calc file called job4621.00.0.calc. Other files will be generated during the correlation process.

4-R3 (1) Information to display

The GUI window for a project should display the following information:

4-R3.1 Name of project

4-R3.2 Jobs

These should be in a list (within a scrollable box if needed) with the following columns:

4-R3.2.1 Name

4-R3.2.2 State

The job state here is the same as that in Req. 3-R4.3.1.1.

4-R3.2.3 Duration

This is the duration associated with this job. It should be displayed in minutes of observe time.

4-R3.2.4 Completion

The fractional completeness (expressed in per-cent) of the job. If the job is not running, this column should be 0% or 100%

4-R3.3 Total duration

The sum of the durations (in minutes) of all the jobs in the project

4-R3.4 Total completion

The fractional completeness of the entire set of jobs, expressed in per-cent). Should be correlated time / total duration.

4-R3.5 (2) Disc space

To be investigated...

4-R4 (1) Actions to support

4-R4.1 Queue All

A button that adds all jobs to the queue should be supported.

4-R4.2 Job highlighting

One or more jobs may be highlighted to select for various actions.

4-R4.3 Add to queue

Selected jobs should be added to queue with the click of a button.

4-R4.4 Delete

Selected jobs will have correlator output files deleted. A warning dialog should verify intent of user. This action will set the state of selected/deleted jobs to one of QUEUED, READY, CONFLICT or WAITING.

4-R4.5 Start Job Manager

If one job is highlighted, the job manager for that job should be started. If multiple jobs are highlighted, the user should be asked if the first one, all, or none of the job managers for the selected jobs should be started.

4-R4.6 (2) Sniff

There should be a button that causes a FITS file to be built and the data to be sent through a quality control analysis program

4-R4.7 View log

A button should all oe the observe log (file ending in vlba.log) to be displayed in a read-only manner in a separate window.

5 Queue Manager

The Queue Manager maintains a list of jobs to be run with some information about the state of those jobs. Note that a single job should not be in the queue in more than one place.

5-R1 (1) Single instance

A single instance of the queue manager should be allowed in one DOI session.

5-R1.1 (2) Possibly the main window
Since there is a 1 to 1 relationship between queue managers and instances of DOIs, it is conceivable that the queue manager can be the root window (launcher of other windows) in the DOI system.

5-R2 (1) Queue state

The queue shall have a state that determines its behavior.

5-R2.1 Queue state

5-R2.1.1 PAUSED
The queue will not start the next job until explicitly told to do so.

5-R2.1.2 RUN FIRST
The first job in the queue will run, but will revert to PAUSED once that job is run.

5-R2.1.3 RUN
All of the jobs in the queue will be run in order. Jobs that are not ready to run (media not available, ...) or jobs that were completed or killed will be skipped.

5-R2.1.4 ERROR
The queue is not active due to an error.

5-R2.2 Default
The queue state should be set to PAUSED upon DOI startup.

5-R2.3 Errors
Upon error running a job, the queue shall enter the ERROR state preventing the running of any other job until the error is acknowledged.

5-R3 (1) Queue Manager Job States

The queue manager, using information from the resource manager, assigns states to particular jobs. These states can be one of:

5-R3.1 QUEUED
The job is in the job queue, but the required media (Mark5 modules) are not loaded.

5-R3.2 READY
The job is ready to run. All required Mark5 modules are loaded and there are no conflicts with other jobs that may be running.

5-R3.3 CONFLICT
Two Mark5 modules for the job are loaded in the same unit, but otherwise the job is ready to run.

5-R3.4 WAITING
The media are all loaded, but one or more Mark5 units containing the media is currently running a different job. WAITING also covers the case where the media are ready, but there is not enough compute resource to start the job.

5-R3.5 RUNNING
The job is running. The job manager window for this job would show additional status information.

5-R3.6 KILLED
The job was stopped prematurely. A partial file exists.

5-R3.7 COMPLETE
The job ran to completion.

5-R4 (1) Queue Manager Display

Each queued job should be listed in a multi-column table where various bits of information for each should be displayed for each. Additionally some overall information should be displayed. The information is not so different from that in the project manager window.

5-R4.1 Per job information
5-R4.1.1 Job name
5-R4.1.2 Queue status of job
5-R4.1.3 Duration of job
Minutes of observe time
5-R4.1.4 Fraction complete
5-R4.2 Global information
5-R4.2.1 The total duration
Minutes of observe time in queue
5-R4.2.2 Remaining duration
Minutes of observe time remaining in queue
5-R4.2.3 Estimated wall time
Estimated amount of wall time remaining, taking into account estimated speed-up factor of projects

5-R5 (1) Actions to allow

The queue manager should allow the following actions to be taken:

5-R5.1 Run
This promotes the selected job to the top of the queue and sets the queue in RUN FIRST state. This will run the selected job and return the queue to the PAUSED state when finished. This should only be an option if a single job is highlighted and no job is currently running.
5-R5.2 Run all
Put the queue in RUN state.
5-R5.3 Pause
Put the queue in the PAUSE state. This will not kill a running job.
5-R5.4 Kill
Kills a running job. This is done by running `stopmpifxcorr` on the head node. The queue state will be set to PAUSED.
5-R5.5 Clear finished
Removes from the queue all jobs that were successfully run. This does not delete any files!
5-R5.6 Purge
Clears all jobs from the queue. This does not delete any files! Any running jobs will not be cleared.
5-R5.7 (2) Queue reordering
There should be a mechanism to reorder the jobs in a queue to run in a specified order.

5-R6 (1) Starting a job

The job start sequence is described here:

5-R6.1 Verify resources
Must go through thre resource manager and verify that needed media (Mark5 modules), Mark5 units, and CPU resources are available.
5-R6.2 Verify input files
Make sure that .input, .calc, .uvw, .delay files exist.

5-R6.3 Job manager
Starts a job manager for the job.
5-R6.4 Claim resources
Mark needed resources as ASSIGNED (see Req.2-R2.1.2) via the resource manager.
5-R6.5 Generate .machines and .threads files
Using resource information, generate the .machines and .threads files.
5-R6.6 Start correlation
Run the `startdifx` script on the designated head node.

5-R7 (1) Queue file

A file shall contain information about the queue.

5-R7.1 Readable
This file is readable by both monitor-mode and control-mode DOIs
5-R7.2 Writable
This file should be writable only by control-mode DOIs
5-R7.3 Editable
The file shall be a text file that is editable by hand if needed.
5-R7.4 Contents
The queue file should contain an ordered list of jobs with the following information:
5-R7.4.1 Job name
5-R7.4.2 Job state
5-R7.4.3 Assigned resources
This is the empty set for jobs that are not running

5-R8 (1) Maintenance

This file should be updated by the control-mode gui every time the state of the queue is changed.