MASSACHUSETTS INSTITUTE OF TECHNOLOGY

HAYSTACK OBSERVATORY

WESTFORD, MASSACHUSETTS 01886

Telephone: *781-981-5400*
Fax: *781-981-0590*

03 Nov 2011

TO: Distribution
FROM: Alan Whitney and David Lapsley
SUBJECT: Mark 6 Command Set (Rev 2.6d)

Revision history:

2.5	12 May 2011	Add 'disks? temp' query to return individual disk temperatures
2.6	21 Jun 2011	Clean-up editing of Section 3, 4 and 5.
2.6a	06 July 2011	Clarifications based on aen comments
2.6b	09 Sep 2011	Change 'remove' command to 'dismount' (more intuitive)
2.6b	09 Sep 2011	Clarify Volume Stack explanation; add figure of example Volume Stack
2.6c		Additional clarifications
2.6d	03 Nov 2011	Add references to physical module slot#'s for operator ease-of-use

Introduction

This memo documents the control program and command set for the Mark 6 VLBI data system when used in normal field recording applications. Different software is normally used to read recording data into the correlator (primarily DiFX software correlators).

1. *dimino6* program

The commands detailed in this memo are implemented by a program called *dimino6* to control the DIM functionality of the Mark 6 VLBI data-recording system. Playback functionality is not handled by this application.

The following procedure is used to start *dimino6*:

TBD

2. Notes on DIM Command set

Note the following with respect to the command set:

1. Processing of all of the commands/queries expect the VSI-S communications protocol and command/response syntax. Later versions will also support *remote procedure calls* (RPCs).
2. Commands/queries are case *insensitive*.
3. Versions of program *dimino6* with a revision date earlier than the date on this memo may not implement all commands indicated in this memo or, in some cases, may implement them in a different way.

3. VSI-S Command, Query and Response Syntax

The following explanation of the VSI-S syntax may be useful in understanding the structure of commands, queries and their respective responses. This explanation has been lifted directly from the VSI-S specification.

3.1 *Command Syntax*

Commands cause the system to take some action and are of the form

<keyword> = <field 1> : <field 2> : …. ;

where <keyword> is a VSI-S command keyword. The number of fields may either be fixed or indefinite; fields are separated by colons and terminated with a semi-colon. A field may be of type decimal integer, decimal real, integer hex, character, literal ASCII or a VSI-format time code. White space between tokens in the command line is ignored, however most character fields disallow embedded white space. For Field System compatibility, field length is limited to 32 characters except for the 'scan label' (see Section 6), which is limited to 64 characters.

3.2 *Command-Response Syntax*

Each command elicits a response of the form

!<keyword> = < VSI return code > :
<Mk6-specific return code> [: <Mk6-specific ASCII return info> : ..] ;

where

<keyword> is the command keyword

<VSI return code> is an ASCII integer as follows:

0 - action successfully completed
1 - action initiated or enabled, but not completed
2 - command not implemented or not relevant to this DTS
3 - syntax error
4 - error encountered during attempt to execute
5 - currently too busy to service request; try again later
6 - inconsistent or conflicting request
7 - no such keyword
8 - parameter error

<Mark6-specific return code> - Mark 6-specific ASCII return code

<Mark6-specific ASCII return info> - optional additional ASCII info relating to Mark 6 return code

3.3 *Query and Query-Response Syntax*

Queries return information about the system and are of the form

<keyword> ? <field 1> : <field 2> : …. ;

with a response of the form

!<keyword> ? <VSI return code> :
[<Mk6-specific return code> [:<Mk6-specific ASCII return info> :] ..] ;

where

<VSI return code> is an ASCII integer as follows:

0 - query successfully completed
1 - action initiated or enabled, but not completed
2 - query not implemented or not relevant to this DTS

3 - syntax error
4 - error encountered during attempt to execute query
5 - currently too busy to service request; try again later
6 - inconsistent or conflicting request
7 - no such keyword
8 - parameter error
9 - indeterminate state

Note: *A 'blank' in a returned query field indicates the value of the parameter is unknown.*
A '?' in a returned query field indicates that not only is the parameter unknown, but that some sort of error condition likely exists.

4. Modules and Volumes

In the Mark 6 system, the set of disks that are recorded simultaneously is referred to as a "volume", where a *volume consists of one or more disk modules*. The following rules apply to volumes:

1. Since disk modules typically hold 8 disks, normal volume sizes are 8, 16, 24 or 32 disks (though there is no restriction on the actual number of disks (up to the limits of the hardware). The maximum number of disks is set by the number of disk controllers installed in the Mark 6 controller (minimum number of supported disks is typically 32).
2. A specified set of disk modules may be 'bonded' to define a multi-module volume; once bonded, that set of disk modules becomes inseparable for purposes of recording or playback until the volume is dissolved by re-initializing the constituent modules.
3. A given disk module may belong to only a single volume (i.e. cannot be shared among different volumes).
4. Multiple volumes may be connected to the Mark 6 controller, but only one may be declared as 'ready' to record at any given time.

Unlike the Mark 5 systems, each Mark 6 module is connected to the Mark 6 controller by a single front-panel e-SATA cable that supports all eights disks of the module. It does not matter which cable is connected to which module; the Mark 6 system identifies modules by interrogating the Module Serial Number (MSN) that is permanently written to each disk in a module when that module is 'initialized'.

5. The Volume Stack

dimino6 creates a logical stack of mounted volumes from which recording resources are allocated; this logical stack is called the 'Volume Stack'. Each time a new module or volume is connected, the Mark 6 identifies the resource and adds it to the Volume Stack; correspondingly, the resource is removed from the Volume Stack when it is logically 'dismounted' and physically disconnected. The Volume Stack operates according to the following rules:

Starting from the top, the Volume Stack is logically divided into two categories (see **Figure 1**):

1. 'active' volumes are defined as ready to record data, and are separated into two sub-categories, in order from the top of the 'active' list:
    a. 'ready' volume: The 'ready' volume is the currently active volume for recording and is always at the top of the Volume Stack; only one volume may be 'ready' at any time.
    b. 'standby' volumes: These volumes are standing by to record data and rise, in order, to the 'ready' position as needed. The topmost 'standby' volume rises automatically to the 'ready' position when the current 'ready' volume becomes full (and hence automatically becomes 'inactive'), is write protected, or is logically 'dismounted' and physically disconnected.

The position of an ‘active’ volume within the Volume Stack may be manipulated using the ‘vol_cmd=’ option ‘up’ (see Command List, Section 9 for details).

2. ‘inactive’ volumes: ‘Inactive’ volumes are, for one reason or another, currently unable to record data; the order of ‘inactive’ volumes in the Volume Stack is inconsequential. A volume may be ‘inactive’ due to:
   - volume is ‘protected’ (i.e. due either to being full or explicitly declared as write-protected)
   - multi-module volume is incomplete (i.e. not all modules of multi-module volume are connected)
   - uninitialized module is connected (i.e. needs to be initialized by the ‘mod_init’ command and assigned a Module Serial Number)
   - operator declares volume ‘inactive’ (for example, in preparation for ‘bonding’ of single modules into a multi-module volume)

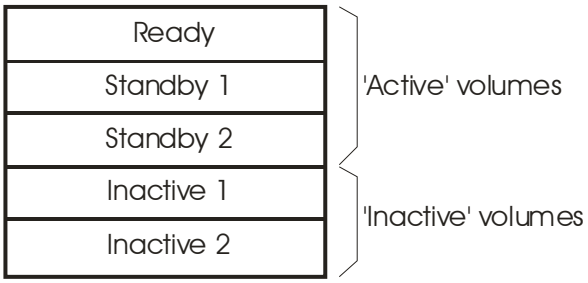

Figure 1: Schematic of example Volume Stack

Moving volumes from ‘inactive’ and ‘active’ status

‘inactive’ volumes may become ‘active’ by the following actions:
- operator declares volume ‘active’ (assuming it otherwise meets the necessary conditions)
- volume ‘protected’ designation is changed to ‘unprotected’
- last module of multi-module volume is connected (making the volume complete)

When an ‘inactive’ volume becomes ‘active’, it automatically becomes a new (and the last) member of the ‘active’ list within the Volume Stack.

Volume References

Each volume within the Volume Stack is automatically assigned a single-character reference (A-Z), in order from the top to the bottom of the Volume Stack. There volume references (‘volrefs’) are used in the command and manipulation of the volumes with the Volume Stack. The following should be noted:
- volrefs are always in alphabetical order and contiguous from the top to the bottom of the Volume Stack
- volref A is always the ‘ready’ volume
- whenever the content or order of the Volume Stack changes, the volref associated with a particular volume may change
- the current Volume Stack may be retrieved at any time with the ‘vol_stack?’ query, which lists all currently mounted modules and their associated volume reference (may be multiple modules per volref)

It may at first seem confusing to allow volrefs to change with changing Volume Stack order, but typically there are no more than 2 or 3 volumes connected at any given time, and the order changes infrequently. A benefit is that the most-referenced volume, the ‘ready’ volume, is always referred to as volref ‘A’. When addressing commands to volumes other than the ‘ready’ volume, it is recommended that you first verify the volref by issuing a ‘vol_stack?’ query. A newly connected module will occupy the last position in the Volume Stack unless that module is a member of an already partially-mounted volume, in which case it will move into a position just below the other mounted modules of that volume. Under normal operating circumstances, a ‘ready’ volume will move up the Volume Stack to the ‘ready’ position as volumes above it in the Volume Stack are filled (and hence become protected and are moved to the bottom of the Volume Stack).

Managing the Volume Stack

The primary command for managing volumes within the Volume Stack is 'vol_cmd='. Options under this command are:

- 'inactive' – declare a volume 'inactive'
- 'active' – declare a volume 'active'
- 'dismount' – prepare the modules of a volume to be disconnected (ceases all activity to volume)
- 'protect' – declare a volume 'protected' to prevent any changes or addition to the data
- 'unprotect'- remove protection to allow volume data to be extended or modified
- 'erase' – erase all data from volume
- 'bond' – bond multiple modules into a single volume
- 'force' – force an incomplete volume to be accepted in lieu of complete volume (for example, when one disk in a volume is dead and you wish to allow recording to the volume on the remaining good disks)
- 'up' – move position of volume in Volume Stack up by one

Details of the 'vol_cmd' are given in Section 9.

6. Scan names, Scan Labels and Linux filenames

dimino6 defines a 'scan' as a continuously recorded set of data. Each scan is identified by a scan name, experiment name and station code, which are normally derived from the information in the associated VEX file used in the scheduling of the experiment (see http://www.vlbi.org).

The filename format for a file containing data from a single scan is typically

<exp name>_<station code>_<scan name>.<file type>

where

<exp name> - experiment name; max 16 chars (consistent with current limit)

<station code> - standard 2-character ASCII station code, or decimal numeric value corresponding to 16-bit numeric station code (see VDIF specification at http://www.vlbi.org for clarification).

<scan name> - assigned scan name (derived from VEX file or other source); max 16 chars

<file type> - identifies high-level data format within file (for example: 'vdif' and 'm5b' for VDIF and Mark5B data formats, respectively)

Maximum scan-label length, including embedded underscores and possible scan-name suffix character, is 50 characters. <exp name>, <stn code> and <scan name> may contain only standard alpha-numeric characters, except '+', '-' and '.' characters may also be used in <scan name>. All fields are case sensitive. No white space is allowed in any of these subfields. Lower-case characters in all subfields are preferred. An example Mark 6 filename is:

grf103_ef_scan123.vdif

In actuality, during the recording of a scan, the Mark 6 writes a single standard Linux file to each disk in the active volume and assigns system-specific unique filenames to each of these files that are different than the prescribed Mark 6 filename. The Mark 6 keeps track of the correspondence between these filenames and the Mark 6 filename through a separate auxiliary files maintained on the data disks; in this way, the group of individual files that comprise a Mark 6 file may be managed and referred to by the Mark 6 filename.

7. *dimino6* Command/Query Summary (by Category)

7.1 System Setup and operation

input_stream	p. 9	Set up input data stream
mod_init	p. 10	Initialize a disk module
record	p. 12	Turn recording on|off
vol_cmd	p. 19	Manage volumes within Volume Stack
vsm	p. 22	Set/get Volume State Mask (VSM): last significant disk operation
vsm_mask	p. 23	Set mask to enable changes in VSM

7.2 Status and information queries

disk_info?	p. 7	Get information about individual disks in specified volumes (query only)
error?	p. 8	Get error number/message (query only)
rtime?	p. 14	Get remaining record time on current volume (query only)
scan_check?	p. 15	Quick check of data in recorded scan (query only)
scan_info?	p. 16	Get summary information for recorded scan (query only)
status?	p. 17	Get system status (query only)
sys_info?	p. 18	Get Mark 6 system information (query only)
vol_stack?	p. 21	Get Volume Stack (query only)

8. *dimino6* Command/Query Summary (Alphabetical)

disk_info?	p. 7	Get information about individual disks in specified volumes (query only)
error?	p. 8	Get error number/message (query only)
input_stream	p. 9	Set up input data stream
mod_init	p. 10	Initialize a disk module
record	p. 12	Turn recording on|off
rtime?	p. 14	Get remaining record time on current volume (query only)
scan_check?	p. 15	Quick check of data in recorded scan (query only)
scan_info?	p. 16	Get summary information for recorded scan (query only)
status?	p. 17	Get system status (query only)
sys_info?	p. 18	Get Mark 6 system information (query only)
vol_cmd	p. 19	Manage volumes within Volume Stack
vol_stack?	p. 21	Get Volume Stack (query only)k
vsm	p. 22	Set/get Volume State Mask (VSM); last significant disk operation
vsm_mask	p. 23	Set mask to enable changes in VSM

9. *dimino6* Command Set Details

This section contains a complete description of all *dimino6* commands/query in alphabetical order.

disk_info – Get disk information (query only)

[command list]

Query syntax: disk_info? [<info_type>] : [<volref>/<slot#>];

Query response: !disk_size? <return code> : <*dimino6* return code> : <type> :
<volref> : <slot#> : <MSN1> : <#disks> : <disk1 info> : …. : <disk*n* info> [:
<volref> : <slot#> : <MSN2> : <#disks> : <disk1 info> : …. : <disk*n* info>] :…;

Purpose: Get information about individual disks in specified volumes

Query parameters:

Parameter	Type	Allowed values	Default	Comments
<info_type>	char	serial | model | size | usage | temp	usage	'serial' – disk serial#'s 'model' – disk model #'s 'size' – disk sizes in GB 'usage' – disk usage in GB 'temp' – disk temperature in degC
<volref> or <slot#>	char int	A-Z (volref) or 1-9 (slot#)	all volumes	Volume reference (alphabetic) or slot# (numeric)

Query response parameters:

Parameter	Type	Values	Comments
<info_type>>	char	-	Information type; default is 'usage'
<MSN1>	char	-	MSN of first (and perhaps only) module in volume
<#disks>	int	-	Number of disks with which module was initialized
<disk*x* info>	-	-	'serial' – disk serial# at initialization; preceded by '-' if disk is missing/dead (see Notes 1 and 2) 'model' – disk model # at initialization (see Note 3) 'size' – disk size in GB at initialization (see Note 3) 'usage' – current disk usage in GB (see Notes 3 and 4) 'temp' – current disk temperature in degC
<MSN2>	real		Extended MSN of second (if it exists) module in volume

Notes:

1. At module initialization, the serial#'s, model#'s and disk sizes of *all* disks in the module are written to *each* disk in the module; this information is then accessible even if one or more of the original disks fail (as long as at least one disk is still OK).
2. For any undiscovered disks (missing/dead), the corresponding reported serial number(s) is preceded by '-'. This allows easy identification of missing/dead disks by serial number.
3. All disk information is listed in the same order as in 'disks? serial' query so that physical disks can be tied to the information.
4. The Mark 6 system writes to disks in a round-robin fashion, but if a disk is not ready to write when its turn comes around, Mark 6 skips over that disk and writes to the next one instead. If this happens repeatedly to the same disk, it will accumulate noticeably less data than its properly operating neighbors. Therefore, if an examination of the amount of data recorded on individual disks across a module reveals significant disparity, one or more slow and/or improperly disks should be suspected. A 'disks? serial;' query may be used to identify corresponding physical disks.

disk_info

error – Get error number/message (query only)

[command list]

Query syntax: error? <*dimino6* return code> ;
Query response: !error ? <return code> : <*dimino6* return code> : <*dimino6*-specific ASCII error message> ;

Purpose: Get *dimino6* error number causing non-zero VSI-S error return

Query parameters:

Parameter	Type	Allowed values	Default	Comments
<*dimino6* return code>	int	-	-	*dimino6* numerical return code

Query response parameters:

Parameter	Type	Values	Comments
<*dimino6* return code>	int		<*dimino6* return code>
<error message>	literal ASCII		Associated ASCII error message, if any

Notes:

1. The 'error?' query is used to retrieve an explanation of *dimino6*-specific error codes.
2. <return code> 8 (parameter error) returned if specified <*dimino6* return code> does not exist.

input_stream – Set up input data stream

[command list]

Command syntax: input_stream = <add|dismount> : <stream_label> : <data format> : <interface_ID> [: <filter address>] ;
Command response: !packet = <return code> : <*dimino6* return code> ;

Query syntax: input_stream? [<stream_label>] ;
Query response: !input_stream? <return code> : <*dimino6* return code> :
<stream_label1> : <data format1> : <interface_ID1> [: <filter address1>] [:
<stream_label2> : <data format2> : <interface_ID2> [: <filter address2>]] :…;

Purpose: Select input data stream
Command parameters:

Parameter	Type	Allowed values	Default	Comments
<add|dismount>	char	add | dismount	-	Specify to 'add' a new input stream definition, or 'delete' to remove an existing definition
<stream_label>	char	16 chars max	-	User-specified label for input data stream (16 chars max); example 'RDBE1'.
<data format>	char	4 chars max	-	'vdif' – VDIF data format; 'm4b' – Mark 5B data format Other data formats may be specified. By default, a lower-case version of this parameter is used as the Mark 6 filename suffix unless overridden by a specification in the 'record=' command. See Note 3.
< interface_ID >	char	16 chars max	-	System-defined interface name (example)
< filter address >	int	IP or MAC address	-	Specify exclusive IP or MAC address from which will be recorded: For IP connection: IP address (i.e. 192.68.1.34) For MAC connection: MAC address (i.e. 01.23.45.67.89.ab; must use '.' as delimiter)

Query parameters:

Parameter	Type	Allowed values	Default	Comments
<stream label>	int	-	All stream labels	Specified stream label

Query response parameters:

Parameter	Type	Values	Comments
<stream_label >	char	16 chars max	User-specified stream label
<data format>	char	4 chars max	Data format
< interface_ID >	char	16 chars max	System-defined interface name
< filter address >	int	IP or MAC address	IP/MAC address for data filtering

Notes:

1. Multiple simultaneous input data streams may be specified (one at a time) and types may be intermixed.
2. For Ethernet connections, entire IP or Ethernet packets are always recorded; during playback, header and/or trailer information around the data frame may be discarded.
3. Each defined 'input stream' is recorded to a separate set of Linux files.

input_stream

mod_init – Initialize a disk module and assign a Module Serial Number (MSN)

[command list]

Command syntax: mod_init = <slot#> : <MSN> : <#disks> ;
Command response: !mod_init = <return code> ;

Query syntax: mod_init? ;

Query response: !mod_init ? <return code> : <*dimino6* return code>
: <slot#> : <extended-MSN1> : <#disks1>
[: <slot#> : <extended-MSN2> : <#disks2> : […] ;

Purpose: Initialize a disk module by assigning it a unique 'permanent' Module Serial Number (MSN).

Command parameters:

Parameter	Type	Allowed values	Default	Comments
<slot#>	int	1-9		Physical slot# of module to be initialized
MSN	char		-	Permanent 8-character 'Module Serial Number' to be assigned to the connected disk module (example:'MPI00153'). MSN format rules are enforced; see Note 2. The module capacity and maximum data rate for the extended-MSN are calculated and appended to the MSN to create the 'extended-MSN' (example 'MPI00153/16/4'); see Notes 1 and 2.
#disks	int	1-16	-	#disks in module. See Note 1

Query parameters: None

Query response parameters: (returns information for all disk modules currently in Volume Stack)

Parameter	Type	Allowed values	Comments
<slot#>	int	1-9	Physical slot#
<extended MSN>	char	-	Example: 'MPI00153/16/4096'; see Notes 2 and 3.
#disks	int	1-16	#disks in associated module

Notes:

1. The 'mod_init=..' command is normally issued only when the module is first procured or assembled, or when the disk configuration is changed. All connected disks in the specified slot# will be initialized; all data on the module will be erased. The number of disks specified must exactly match the number of disks discovered by the Mark 6 controller in the specified slot#. An error will be returned if #disks specified is different from number of disks discovered by the Mark 6 controller.
2. The format of the **Module Serial Number** (MSN) is enforced as follows:
   MSN – Must be 8 characters in length with format 'ownerIDserial#' for Mark 5 modules converted for use with Mark 6, or 'ownerID_serial#' for XCube modules, where:

a. ownerID – 2 to 6 upper-case alphabetic characters (A-Z). The 'ownerID' must be registered with Jon Romney at NRAO (jromney@nrao.edu) to prevent duplicates. Numeric characters are not allowed. Any lower-case characters will automatically be converted to upper case.
b. serial# - numeric module serial number, with leading zeroes as necessary to make the MSN exactly 8 characters long. Alphabetic characters are not allowed in the serial#.

Note: The MSN is entirely equivalent to the 'VSN' assigned to Mark 5 modules; both are serial numbers assigned to disk modules. The 'V' in 'VSN' stands for 'volume', but is totally unrelated to the Mark 6 definition of 'volume'; and is an anachronistic relic of long gone tape systems.

3. '**Extended-MSN**' is of the form '<MSN>/<total TB>/<disk type>'. *dimino6* queries the disks to compute the capacity of the module in TB and disk models; if mixed disk models, <disk type> will be set to "mixed"; these fields are then appended to the MSN to create the extended MSN. Mixed disk models and sizes are supported by the Mark 6, but not encouraged, particularly mixed disk sizes.
4. MSN? query will return information for all connected disk modules. If number of disks identified by the MSN? query differs from the number of 'registered' disks, an error will result.
5. The replacement of any disk(s) in an initialized module requires re-initialization of the module with the 'mod_init=' command before data can be recorded. All existing data will be lost.

mod_init=

record – Turn recording on|off; assign scan label

[command list]

Command syntax: record = <on|off|time> : [<start time>] : [<duration>] : [<data size>] : [<scan name>] : [<experiment name>] : [<station code>]:
;

Command response: !record = <return code> : <*dimino6* return code> ;

Query syntax: record? ;

Query response: !record ? <return code> : <*dimino6* return code> : <status>: <scan number> : <scan name> ;

Purpose: Turn recording on|off or specify UT start time.

Command parameters:

Parameter	Type	Allowed values	Default	Comments
<on|off|time>	char	on | off | time		'on' immediately starts recording; 'off' stops recording; 'time' specifies <start time> and <duration>. Error will be returned if specified <start time> has passed; 'record=<off>' command will abort 'time' recording (either before or after recording is started). See Notes 1-4.
<start time>	time			For 'time' mode: scheduled UT recording start time; VEX time format See Notes 4&5.
<duration>	int	seconds		If specified, recording will stop after this amount of time; 'record=off' command may be used to stop recording before specified duration is complete.
<data size>	real	-	-	Total amount (GB) of data expected to be recorded in this scan; used to judge whether sufficient space exists on present 'ready' volume. See Note 3.
<scan name>	char	32 chars max		Relevant only for 'record=on' command. If not specified, the current Mark 6 system UT time will be used to create a scan name in the format 'scan_xxxx', where 'xxxx' is a Mark6 generated scan serial number
<experiment name>	char	8 chars max	expxx	Relevant only for 'record=on' command; once specified, <experiment name> will be remembered on subsequent 'record=on|time' commands; 'record=on|time' with <experiment name> will reset default.
<station code>	char	8 chars max	stnxx	Relevant only for 'record=on' command; once specified, <station name> will be remembered on subsequent 'record=on|time' commands; 'record=on|time' with <station name> will reset default.

Query parameters: None

Query response parameters:

Parameter	Type	Values	Comments
<status>	char	off | pending | recording | flushing	'off' – recording currently inactive 'pending' – awaiting timed start 'recording' – capturing data to RAM and recording to disk 'flushing' – flushing RAM buffers to disk after data capture has stopped
<scan number>	int		Mark6-assigned scan serial number
<scan name>	char		Scan name –see Section 6 for definition of scan name.

record

Notes:

1. Recording always take place to the 'ready' volume (volref 'A') at the top of the Volume Stack.
2. During recording, commands to the system that may interfere with the recording capability of the disks (such as queries for MSN, disk models, serials numbers, directory list) will be rejected.
3. If the <data size> parameter is specified, *dimino6* will first check the present 'ready' volume for sufficient space. If insufficient space, *dimino6* will 'protect' the volume, causing it to move to the 'inactive' part of the Volume Stack; the first 'standby' volume will become the new 'ready' volume on which the recording takes place; if no 'standby' volume is available, an error will be returned. During or after the recording, the 'scan_info?' query may be used to determine which volume was selected for recording, plus other scan information.
4. Full VEX time format is '##y###d##h##m##ss.ss', but may be truncated from left or right as long as it is ambiguous at the time it is issued (e.g. '30m10s' will start at specified time within the next hour).
5. While waiting for 'time' recording to start, status of 'ready' volume is not allowed to change.
6. For recording performance reasons, once a scan is recorded, there is no mechanism to delete it other than to erase the entire volume.

record

rtime – Get remaining recording time on current volume (query only)

[command list]

Query syntax: rtime? [<volref>] [<data rate>];
Query response: !rtime ? <return code> : <*dimino6* return code> :
<volref> : <data rate> : <remaining time> : <remaining GB> : <remaining percent> ;

Purpose: Get remaining recording time of active volume for specified data rate.

Query parameters:

Parameter	Type	Allowed values	Default	Comments
<volref>	char	A-Z	A	Volume reference
<data rate>	real	Gbps	1 Gbps (not recording); estimated rate (recording)	If actual data rate different than default, <remaining time> response will need to be scaled proportionally.

Query response parameters:

Parameter	Type	Values	Default	Comments
<data rate>	real	Gbps	1.0	Recording rate assumed in calculation of <remaining time>; default 1Gbps
<remaining time>	int	seconds	-	Approximate remaining record time for currently 'active' volume at specified (or default) data rate.
<remaining GB>	int	GB		GB remaining on current 'active' volume
<remaining percent>	int	-		Remaining percentage of disk space still available

Notes:

1. If an 'rtime?' query is issued during recording and <data rate> is not specified, the Mark 6 system estimates the actual aggregate recording data rate and returns the <remaining time> estimate accordingly.

scan_check – Quick check of recorded data (query only)

[command list]

Query syntax: scan_check? [<volref>] : [<scan name|scan#>] ;

Query response: !scan_check ? <return code> : <*dimino6* return code>: <volref> : <scan#> : <scan label> : <#streams>
: <stream label1> : <status1> : <data format1> : <start time1> : <duration1> : <datasize1> : <stream rate1>
: <stream label2> : <status2> : <data format2> : <start time2> : <duration2> : <datasize2> : <stream rate2> : …
[: <stream labeln> :…: <stream raten>] ;

Purpose: Quick-look check of recorded data from completed scan

Query parameters:

Parameter	Type	Allowed values	Default	Comments
<volref>	char	-	A	Volume reference
<scan name|scan#>	char/int	-	Last recorded scan	Default is last scan. See Note 2.

Query response parameters:

Parameter	Type	Values	Comments
<volref>	char	A-Z	Volume reference
<scan#>	int		Mark6-assigned sequential scan# within the volume
<scan label>	char	-	Scan label (see Section 6)
<#streams>	int	-	#streams of recorded data
<stream labelx>	char	-	Stream label as defined by ‘input_stream=’ command
<statusx>	char	OK | time? | data?	‘OK’ – data frames and data appear to be normal ‘time?’ – error decoding time in data frames ‘data? – sampled data do not appear to be statistically random; see Note 5
<data formatx>	char	-	Data format (‘vdif’, ‘m5b’, etc)
<start timex>	time	UT time	Time tag decoded from first data-frame in file, in format ##y###d##h##m##ss
<durationx>	time	seconds	Decoded time interval between first and last data-frames of scan
<data sizex>	real	GB	Total data recorded (GB) from data streamx; see Note 6
<stream data ratex>	real	Gbps	Inferred stream data rate (Gbps) based on starting/ending time tags and <file size>; see Note 6.

Notes:

1. ‘record?’ status must be ‘off’ when doing scan_check; attempting scan_check during recording will return an error.
2. ‘scan#’ is the Mark6-assigned sequential scan# within the volume.
3. <data format> is taken from scan metadata recorded in the volume directory; supported suffixes are ‘vdif’ and ‘m5b’ (plugins may be developed to support other data formats).
4. ‘scan_check’ examines a small amount of data near the scan start and stop points; the data are decoded according to the specified format.
5. Statistical randomness of data is based on a small sample of data at both beginning and end of scan; not authoritative, but should be looked at.
6. Mark 6 records entire Ethernet frames, which is taken into account in calculating <stream data rate>.

scan_check

scan_info – Get scan information (query only)

[command list]

Query syntax: scan_info? [<volref>] : [<scan name|scan#>] ;

Query response: !scan_info ? <return code> : <*dimino6* return code> : <volref> : <Mark 6 S/N>
<scan#> : <scan label> : <status> : <start time> : <duration> : <#data streams>
: <MSN1> : <#disks> : <disk1datasize> : ….: <diskndatasize>
[….: <MSNn> : <#disks> : <disk1datasize> : : <diskndatasize>] : ….;

Purpose: Get summary information on recorded scan

Query parameters:

Parameter	Type	Allowed values	Default	Comments
<volref>	char	-	A	Volume reference
<scan name|scan#>	char/int	-	Last recorded scan	Default is last scan. See Note 2.

Query response parameters:

Parameter	Type	Values	Comments
<volref>	char	A-Z	Volume reference
<Mark 6 S/N>	char	-	Serial of Mark 6 unit on which data were recorded
<scan#>	char	-	Mark6-assigned sequential scan# within the volume
<scan label>	int	-	Scan label (see Section 6)
<status>	char	recording | pending | flushing | complete	‘recording’ – capturing data to RAM and recording to disk ‘pending’ – awaiting timed start ‘flushing’ – flushing RAM buffers to disk after data capture has stopped ‘compete’ – all data recorded to disk
<start time>	time	UT time	Recording UT start time in format ‘##y###d###h##m##ss’, where ‘#’ is a digit.
<duration>	int	seconds	Recording duration in seconds
<#data streams>	int	-	# of data streams in scan; corresponds to # of files per disk (one file per data stream)
<MSNn>	char	8 chars	Module Serial Number of first module in volume; order of MSNs in volume is immaterial.
<#disks>	char	int	#disks in MSNn
<diskxdatasize>	char	real	Total recorded data (GB) for this scan on diskx of MSNn, reported in order of ‘disk_info? serial’ query

otes:

1. The ‘scan_info?’ query may be issued at any time (even during active recording).
2. ‘scan#’ is the Mark6-assigned sequential scan# within a volume.
3. A primary use of ‘scan_info?’ is to monitor relative disk performance across all disks in a volume by returning the amount of data written on each disk in the volume for the target scan; can be used in conjunction with ‘disk_info? usage’ query to get good overall view of individual disk performance.

scan_info

status – Get system status (query only) [needs definitions for Mark 6]

[command list]

Query syntax: status? ;
Query response: !status ? <return code> : <*dimino6* return code>: <status word> ;

Purpose: Get general system status.

Query parameters: None

Query response parameters: (TBD by software author)

Parameter	Type	Values	Comments
<status word>	hex	-	Bit 0 – (0x001) Bit 2 – (0x0004) Bit 3 – (0x0008) --- Bit 4 – (0x0010) Bit 5 – (0x0020) Bit 6 - (0x0040) Bit 7 - (0x0080) --- Bit 8 - (0x0100) Bit 9 - (0x0200) Bit 10 – (0x0400) Bit 11 – (0x0800) --- Bit 12 – (0x1000) Bit 13 – (0x2000) Bit 14 – (0x4000) Bit 15 – (0x8000) --- Bit 16 – (0x10000) Bit 17 – (0x20000) Bit 18 – (0x40000) Bit 19 – (0x80000) --- Bit 20 – (0x100000) Bit 21 – (0x200000) Bit 22 – (0x400000) Bit 23 – (0x800000) --- Bit 24 – (0x1000000) Bit 25 – (0x2000000) Bit 26 – (0x4000000) Bit 27 – (0x8000000)

status

sys_info – Get system information (query only)

[command list]

Query syntax: sys_info? ;

Query response: !sys_info ? <return code> : <*dimino6* return code> :
<system type> : <Mark 6 S/N> : <OS type/rev> : <*dimino6* version number> :
<command set revision> : <available RAM> :<#data disks supported> :
<#Ethernet input ports> : <portref1> : <portspeed1> :…. : <portref*n*> : <portspeed*n*> ;

Purpose: Get Mark 6 system information

Query parameters: None

Query response parameters:

Parameter	Type	Values	Comments
<system type>	char	Mark6	
<Mark 6 serial number>	char		Assigned system serial number; generally in the form '??????'
<OS type/revision>	char		example: 'Ubuntu 10.10'
<*dimino6* version number>	char		Version number of *dimino6*
<command set revision>	char		Mark 6 DIM command set revision level corresponding to this software release (e.g. '1.1')
<available RAM>	int		Available burst-mode RAM (GB)
<#data disk supported	int		max# data disks supported on system
<#Ethernet input ports>	int		#Ethernet ports available for data input
<portref*x*>	char		Port reference name to be used in 'input_stream" command (e.g. 'eth0')
<portspeed*x*>	int	1 | 10	Nominal Ethernet speed (Gbps) (e.g. '1' or '10')

sys_info

vol_cmd – Manage volumes within Volume Stack

[command list]

Command syntax: vol_cmd = <action> : <volref1> : [<volref2> :] ;
Command response: !vol_cmd = <return code> : <*dimino6* return code> ;

Query syntax: vol_cmd? ;
Query response: !vol_cmd ? <return code> : <*dimino6* return code>: <vol_ref1> : <vol_ref2 > [: …] ;

Purpose: Manage volumes within Volume Stack.

Command parameters:

Parameter	Type	Allowed values	Default	Comments
<action>	char	inactive | active | dismount | protect |unprotect | erase | bond | force | up	-	See Notes and state-transition table below
<vol ref>	char	A | B |…	-	Target volume reference(s), if required by option

Query parameters: None

Query response parameters:

Parameter	Type	Values	Comments
vol_ref1, vol_ref2, etc	char	A | B | (etc)	List of attached volumes references

Rules:

a. Multiple simultaneously connected volumes are allowed; only one volume may be 'ready'(enabled to record) at any given time, and it is always at the top of the Volume Stack and defined as volume reference 'A'.
b. An arbitrary number of 'standby' volumes may follow the 'ready' volume in the Volume Stack, followed by any 'inactive' (i.e. not ready for recording) volumes.

The following table shows allowed volume state transitions:

Current status ↓	Requested action: Inactive[4]	Active[5]	Dismount[6]	Protect[7]	Unprotect[8]	Erase[9]	bond[10]	force[11]	up[12]
recording	x	x	x	x	x	x	x	x	x
ready	inactive	null	(dismounted)	x	null	x	x	x	null
standby	inactive	null	(dismounted)	x	null	x	x	x	stand by or read y
inactive[1]	null	active	(dismounted)	protected	inactive	inactive	inactive	x	x
partial_vol[2]	x	x	(dismounted)	x	x	x	x	active	x
uninitialized[3]	x	x	x	x	x	x	x	x	x
protected	x	x	(dismounted)	-	active	standby	x	x	x

Table 1 : State transition table; 'x'=not allowed (see Notes corresponding to superscripts)

vol_cmd

Notes:

1. ‘inactive’ volumes include any volumes not available to record, including ‘partial-vol’, ‘uninitialized’ and ‘protected’ status. Order of ‘inactive’ volumes in Volume Stack is inconsequential.
2. A ‘partial_vol’ may indicate a volume with fewer discovered disks than nominal (i.e. all constituent modules not connected, or a failed disk in one or modules of the volume). If a discovered volume becomes complete, the volume status will automatically change from ‘partial_vol’ to whatever is appropriate for the condition of the volume (e.g. ‘ready’, ‘standby’, ‘protected’, etc.). A ‘force’ command may be used to force acceptance of a partial volume (see Note 11).
3. An uninitialized module must be initialized with the ‘mod_init=’ command (assigns Module Serial Number and erases all data).
4. A ‘ready’ volume that is moved to ‘inactive’ will move that volume to the top of the ‘inactive’ section of the Volume Stack; the first ‘standby’ volume in the Volume Stack will move up to ‘ready’. A ‘standby’ volume moved to ‘inactive’ will move to the top of the ‘inactive’ section of the Volume Stack.
5. An ‘inactive’ volume set to ‘active’ (i.e. ‘ready’ or ‘standby’) will be re-positioned as the last ‘active’ volume in the Volume Stack.
6. A ‘dismounted’ volume is deleted from the Volume Stack and may be physically disconnected.
7. A ‘ready’ or ‘standby’ volume must be declared ‘inactive’ before it can be protected.
8. A ‘ready’ or ‘standby’ volume must already be ‘unprotected’ by definition.
9. A volume must be ‘inactive’ before erasure. Request to ‘erase’ a volume requires a ‘vol_cmd=unprotect’ command immediately preceding the ‘vol_cmd=erase’ request. An uninitialized module may be erased by the ‘msn’ command as part of the initialization process. The module will move to a position immediately below all existing ‘ready’ and ‘standby’ volumes in the Volume Stack.
10. Modules must be empty, unprotected and ‘inactive’ before bonding. Once modules are bonded into a volume, the bonding may be dissolved only by re-initializing the individual constituent modules with a ‘mod_init=’ command.
11. The ‘vol_cmd=force:volref;’ command forces a partial volume to be accepted in lieu of the nominal complete volume (e.g. forces a volume to be accepted with only 7 of 8 disks working); in all other respects, the volume behaves as an ordinary volume, though the capacity and maximum data rate of the affected module will obviously be impacted.
12. ‘up’: Moves designated volume ‘up’ one position in Volume Stack; cannot move ‘inactive’ volume into ‘active’ area of Volume Stack; cannot move ‘standby’ module into ‘ready’ position while recording. Order of ‘inactive’ volumes (‘partial_vol’, uninitialized’, ‘protected’) in Volume Stack has no functional consequences.

vol_cmd

vol_stack – Get current Volume Stack (query only)

[command list]

Query syntax: vol_stack? [<volref>] ;

Query response: !vol_stack? <return code> : <*dimino6* return code> :
<volref> : <slot#> : <exMSN1> : <#disks nominal> : <#disks discovered> : <%full> : <module status>
: [<volref>.: <slot#> : <exMSN2> : <#disks nominal> : <#disks discovered> " <%full> : <module status>
[:…] ;

Purpose: Get current Volume Stack information

Query parameters:

Parameter	Type	Allowed values	Default	Comments
<vol_ref>	char	A-Z	All volumes	Optional: target volume reference

Query response parameters (returns following information for each module of each volume, unless single volume is specified)

Parameter	Type	Values	Comments
<vol_ref>	char	A-Z	Volume reference character (A-Z)
<slot#>	int	0-9	Physical slot#; see Note 3
<exMSN>	char	-	Extended Module Serial Number (see Notes for 'mod_init' command)
<#disks nominal>	int	-	Number of disks expected
<#disks discovered>	int	-	Number of disks in module discovered
<%full>	-	int%	Percent that module is filled with data
<module status>	char	recording | ready | standby | inactive | partial_vol | missing | uninitialized | protected	For multi-module volumes, <module status> will be the same for all modules within volume.

Notes:

1. By default, returns monitor parameters for each module of all attached modules, as well as missing modules of multi-module volumes.
2. Example Volume Stack from 'vol_stack?;' query:

0 :0 : A : 1 : MPI00243/16/4096 :8 : 8 : 21% : ready :	Single-module Volume A with 8 disks is ready for recording
B : 2 : MPI00250/8/2048 : 8 : 8 : 0% : standby :	2-module Volume B in slots 2 and 3; 'standby' status
B : 3 : MPI00251/8/2048 : 8 : 8 : 0% : standby :	
C : 4 : MPI0277/16/4096 : 8 : 8 : 0% : protected ;	'protected' single-module Volume C is protected (and inactive)

3. Slot# returned =0 for 'missing' module of multi-module Volume

vol_stack

vsm –Set/get Volume State Mask (VSM): last significant volume operation

[command list]

Command syntax: VSM = <volref> : <VSM> ;
Command response: !VSM = <return code> : <*dimino6* return code>: <VSM> ;

Query syntax: VSM? <volref>;
Query response: !VSM? <return code> : <*dimino6* return code> :
<volref1> : <MSN1a[/MSN1b]> : <VSM1> : <volref2> : <MSN2> : <VSM2> […] ;

Purpose: Set/get Volume State Mask (VSM): logs the last significant volume operation.

Command parameters:

Parameter	Type	Allowed values	Default	Comments
<volref>	char	A-Z	All volumes	
<VSM>	char	recorded | played | erased |unknown | error	none	To be used only if automatically-set VSM parameter is to be overwritten. Requires a preceding 'vol_cmd= <protect>:<volref>; command. Current value of VSM is ignored.

Query parameters

Parameter	Type	Allowed values	Default	Comments
<vol_ref>	char	A-Z	All volumes	Target volume reference

Query response parameters:

Parameter	Type	Values	Comments
<volref>	char	A | B | etc	Volume references
<MSN>	char	-	Module(s) serial number
<VSM>	char	recorded | played | erased | unknown | error	recorded – last significant operation was record or a record-like function played – last significant operation was playback erased – last significant operation was erase unknown – ? error – error occurred; for example, a failure during one of the significant operations above

Notes:

1. Normally, the setting of the VSM parameter happens automatically whenever a record, play or erase command is issued. However, the <VSM=...> command is provided to manually overwrite the current VSM parameter. This command requires a preceding 'protect=off' and affects only the active module. A 'VSM=…" command ignores the current value of the VSM (see 'VSM_mask' command).
2. The VSM logs the last significant volume operation. It is designed to distinguish between data volumes waiting to be correlated, have been correlated, or have no data (erased) and ready to be recorded. The VSM is saved on the volume disk module(s) in the same area as the permanent MSN. The scan_check command does not affect VSM.
3. The 'VSM' and 'VSM_mask' commands were requested by NRAO and are designed primarily for use at a correlator.
4. If no modules are inserted, an error code 6 is returned.

VSM

vsm_mask – Set mask to enable changes in Volume State Mask (VSM)

[command list]

Command syntax: VSM = <erase_mask_enable> : <play_mask_enable> : <record_mask_enable>;
Command response: !VSM = <return code> : <*dimino6* return code> :
<erase_mask_enable> : <play_mask_enable> : <record_mask_enable> ;

Query syntax: VSM? ;
Query response: !VSM? <return code> : <*dimino6* return code> :
<erase_mask_enable> : <play_mask_enable> : <record_mask_enable> ;

Purpose: Set mask to enable changes in Volume State mask (VSM).

Command parameters:

Parameter	Type	Allowed values	Default	Comments
<erase_mask_enable>	int	0 | 1	1	0 – disable an erase operation from modifying the VSM. 1 – enable erase operation to modify the VSM.
<play_mask_enable>	int	0 | 1	1	0 – disable a play operation from modifying the VSM. 1 – enable play operation to modify the VSM.
<record_mask_enable>	int	0 | 1	1	0 – disable a record operation from modifying the VSM. 1 – enable record operation to modify the VSM.

Query parameters: None

Query response parameters:

Parameter	Type	Values	Comments
<erase_mask_enable>	int	0 | 1	
<play_mask_enable>	int	0 | 1	
<record_mask_enable>	int	0 | 1	

Notes:

The VSM mask is intended to prevent accidental changes in the VSM. When a module is at a station, the VSM setting of 1:0:1 will disable a play operation from modifying the VSM. Likewise, at a correlator one might want to disable the record_mask_enable.

VSM_mask