**Mark 6 usage examples** (Rev 2.6d)
arw 2 Dec 2011
Note: only selected responses shown

**Example 1: Initialize new module**

```shell
vol_stack? ;
!vol_stack? 0:0:1:8::uninitialized;
mod_init = 1: HAYS0001:8;
vol_stack?;
!vol_stack? 0:0:A:1: HAYS0001/16/4096:8:8:0%:ready;
```

*Get Volume Stack; uninitialized module in Slot 1

*Initialize, assign MSN, erase all data on module

*Module assigned to Vol A

*and is ready to record; 0% full

*Mount module in Slot 2 that is protected (because it is full); erase module in preparation for recording

```shell
vol_stack?;
!vol_stack? 0:0:A:1: HAYS0001/16/4096:8:8:0%:ready;
```

*Already mounted in Example 1

*B: 2: HAYS0002/16/4096:8:8:99%:protected; 2nd module assigned Volume B

*Unprotect in preparation for erase

*Erase all data on volume

*Vol ref A ready to record

*Vol ref B on standby

**Example 2: Record a scan and do a quick check of the data**

```shell
input_stream = add: RDBE1: vdir: eth0: 192.162.1.38;
input_stream = add: RDBE2: vdif: eth0: 192.162.1.40;
record = on: 076-1233: exp123: wf;
vol_stack?;
!vol_stack? 0:0:A:1: HAYS0001/16/4096:8:8:0%:recording:
```

*Define 1st input data stream, source, data format, and specify IP filter

*Define 2nd data-input stream

*Start recording scan on volume A

*Vol ref A recording

*at schedule end of scan....

```shell
record = off;
```

*Stop recording

```shell
scan_info?;
```

*Get summary scan info

```shell
!scan_info? 0:0:A:Mk6-025:1:076-1233_exp123_wf:complete:2011h076d12h33m00s:80:2:
```

*Disks are performing uniformly

```shell
HAYS0001:8.0:5.0:5.0:5.0:5.0:5.0:5.0:5.0:
```

*Vol ref A recording

*Stop recording

```shell
scan_check?;
```

*Do quick data sanity check

```shell
!scan_check? 0:0:A:1:076-1233_exp123_wf:2:
```

*RDBE1: OK: vdif:2011h076d12h33m01s:79.9:40.0:4.0:

*RDBE2: OK: vdif:2011h076d12h33m01s:79.9:40.0:4.0;
Example 3: Start recording with insufficient space left on ‘ready’ volume.

vol_stack?;
!vol_stack? 0:0: A : 1 : HAYS0001/16/4096 : 8 : 8 : 99% : ready: Vol ref A ready (but nearly full) : B : 2 : HAYS0002/16/4096 : 8 : 8 : 0% : standby ; Vol ref B on standby

record = on : 076-1330 : exp123 : wf : 200; Start recording; estimated size 200GB

scan_info?;
!scan_info? 0:0: B : Mk6-025 : 1 : 076-1330_exp123_wf : recording : 2011h076d13h30m:00s : 24 : 2 : HAYS0002 : 8 : 3.0 : 1.6 : 3.0 : 3.0 : 3.0 : 3.0 : 3.0 ; Disk 2 appears to be slow

Note recording has been switched to new volume (HAYS0002) since original ‘ready’ volume had insufficient space. Querying the Volume Stack shows what happened.

vol_stack?;
!vol_stack? 0:0: A : 1 : HAYS0002/16/4096 : 8 : 8 : 0% : recording: ‘Standby’ volume now at top of Volume Stack : B : 2 : HAYS0001/16/4096 : 8 : 8 : 99% : protected; Full volume B now ‘protected’ and inactive

vol_cmd = dismount : B ; Dismount full volume B from vol stack

vol_stack?;
!vol_stack? 0:0: A : 1 : HAYS0002/16/4096 : 8 : 8 : 0% : recording; Volume A continues to record

Example 4: Bond two modules into single ‘volume’ and start recording

Connect two modules that you want to ‘bond’ into a single volume.

vol_stack?;
!vol_stack? 0:0: A : 3 : HAYS0003/16/4096 : 8 : 8 : 97% : ready: First module in Slot 3 : B : 4 : HAYS0004/16/4096 : 8 : 8 : 76% : standby; Second module in Slot 4; Slots 1 & 2 empty

Modules must both be empty and ‘inactive’ before bonding.

vol_cmd = inactive : A : B ; Make vols A and B ‘inactive’

vol_cmd = unprotect : A : B ; Must ‘unprotect’ command immediately before erase

vol_cmd = erase : A : B ; Erase all data on both modules

vol_stack?;
!vol_stack? 0:0: A : 3 : HAYS0003/16/4096 : 8 : 8 : 0% : inactive: Module inactive : B : 4 : HAYS0004/16/4096 : 8 : 8 : 0% : inactive; Module inactive

vol_cmd = bond : A : B ; Bond A and B into single volume

vol_stack?;
!vol_stack? 0:0: A : 3 : HAYS0003/16/4096 : 8 : 8 : 0% : standby : Modules are ‘bonded’ into volume A and automatically added to ‘standby’ list : A : 4 : HAYS0004/16/4096 : 8 : 8 : 0% : standby;

record = on :......... Start recording

vol_stack?;
!vol_stack? 0:0: A : 3 : HAYS0003/16/4096 : 8 : 8 : 0% : recording: Recording to 2-module volume A : A : 4 : HAYS0004/16/4096 : 8 : 8 : 0% : recording;
Example 5: Only one of a pair of ‘bonded’ multi-module volume is connected

```
vol_stack?;
!vol_stack? 0:0: A : 1 : HAYS0005/16/4096 : 8 : 8 : 52% : partial_vol:  
: A : 0 : HAYS0006/16/4096 : 8 : 0 : 0 : missing;  
```

Only partial volume connected
The missing module is identified;
Slot# returned as 0

Connect the missing module

```
vol_stack?;
!vol_stack? 0:0: A : 1 : HAYS0005/16/4096 : 8 : 8 : 52% : ready:  
: A : 3 : HAYS0006/16/4096 : 8 : 8 : 52% : ready;  
```

2-module volume A ready to record;
missing module installed in Slot 3

Example 6: Force 8-disk module with missing/dead disk to be made ready for recording; identify missing/dead disk

Connect the module. Undiscovered disk (perhaps bad) reduces disk count to 7 and causes ‘partial_vol’ status

```
vol_stack?;
```

Only 7 disks discovered!

Might be good idea at this point to ‘dismount’ module and re-seat data cables, or connect with new data cable(s) to eliminate bad connection as cause of partial volume. Or you may force acceptance of 7 disks as follows:

```
vol_cmd = force: A ;  
```

Force acceptance of partial volume; data preserved

```
vol_stack?;
```

2-module volume A ready to record
(albeit with diminished data and data-rate capacity)

Identify serial number of missing/dead disk

```
disk_info? serial;  
```

SN3 is missing/dead

Example 7: Check disks for uniformity of usage; find slow disk

```
vol_stack?;
!vol_stack? 0:0: A : 1 : HAYS0008/16/4096 : 8 : 67% : ready;  
```

Module is ready for recording

Get disk-by-disk usage (GB) with volume (one module in this case)

```
disk_info? usage;  
```

5th disk appears slow
5th disk in list has much less data than other (152GB vs ~699GB on other 7 disks), so is likely slow.

```
disk_info? serial;  
```

SN5 is serial# of slow disk; we can now uniquely identify the slow disk and replace it.

3