SGI Media Server for Broadcast Reference Pages

- mcclips - MSB clip manager
- mccompstats - MSB compression monitor
- mcpanel - MSB media control panel
- mcstat - MSB status display
- mvcp - Multiport Video Computer Protocol
- rotatelogs - Rotate MSB log files
- VST - Video Server Toolkit
- vst-controls - Media Server For Broadcast (MSB) Controls
- vtrclip - MSB clip insertion/deletion utility
- vtrd - MSB parent daemon
- vtrdircopy - Directory Copy Daemon for MSB
- vtrfsinfo - Retrieve MSB filesystem manifest
- vtrftp - Command Line Ftp Client for MSB
- vtrftpd - Real-time enabled FTP daemon
- vtrhwinfo - Retrieve MSB hardware manifest
- vtrstart - MSB startup
- vtrstat - MSB status
- vtrstop - MSB shutdown
- vtrswinfo - Retrieve MSB port status
- vtrswinfo - Retrieve MSB software manifest
- vtrsyncinfo - MSB video sync information tool
- vtrvfutil - MSB vframe clip utility
- vvtr - MSB process

mcclips(1)

NAME

mcclips - MSB clip manager

SYNOPSIS

/usr/vtr/bin/mcclips [**-v** *level*] [[**-h**] *hostname*]

DESCRIPTION

mcclips provides basic functions for querying and managing the clips stored on a MSB.

mcclips has the following options:

-h *hostname*
: Mangage the clips on the MSB *hostname*. If this option is not specified, the local host is assumed.

-v *level*
: Sets the logging verbosity to *level*.

SEE ALSO

mcpanel(1), *mcstat*(1)

mccompstats(1)

NAME

mccompstats - MSB compression monitor

SYNOPSIS

/usr/vtr/bin/mccompstats [**-v** *level*] [[**-h**] *hostname* [[**-p**] *port*]]

DESCRIPTION

mccompstats displays video compression ratio for a clip as it is played by the MSB *hostname* out the video *port*.

mccompstats has the following options:

-v *level*
Sets the logging verbosity to *level*.

SEE ALSO

mcstat(1)

mcpanel(1)

NAME

mcpanel - MSB media control panel

SYNOPSIS

/usr/vtr/bin/mcpanel [**-h** *hostname*] [**-p** *port*] [**-D** *deck-port*]
[**-c** *clip*] [**-r**] [**-C** "*in out*"] [**-P**]
[**-v** *level*] [**-o** *owner-info*]

DESCRIPTION

mcpanel is a VTR emulation application for MSB.

mcpanel has the following options:

-c *clip*
Load the clip named *clip*. If this option is not specified, no clip is initially loaded.

-D *deck-port*
Specifies that a deck will be controlled using the deck control port named *deck-port*. If this option is not specified, deck control is not available.

-h *hostname*
Contact the MSB running on *hostname*. If this option is not specified, the local host is assumed.

-p *port*
Specifies that the control panel accesses the video port named *port*. If this option is not specified, the first video port on the MSB is used.

-r
Specifies that if an mcpanel already exists for the video port, that mcpanel should be raised on the desktop instead of creating a new mcpanel.

-C "*in out*"
Cues the loaded clip using the *in* and *out* points. Specify "*" for either *in* or *out* to use the default mark in/out.

-o *owner-info*
Insert in mcpanel title bar the *owner-info*. Maximum character size of *owner-info* is 50 characters.

-P
Starts playing the loaded clip.

-v *level*
Sets the logging verbosity to *level*.

SEE ALSO

mcclips(1), *mcstat*(1)

mcstat(1)

NAME

mcstat - MSB status display

SYNOPSIS

/usr/vtr/bin/mcstat [**-v** *level*] [[**-h**] *hostname*]

DESCRIPTION

mcstat displays the current status of all the ports and units currently open on a MSB.

mcstat has the following options:

-h *hostname*
Display the status of the MSB *hostname*. If this option is not specified, the local host is assumed.

-f *framerate*
sets the framerate. *framerate* may be 24, 25, 29.97 or 30.

-v *level*
Sets the logging verbosity to *level*.

SEE ALSO

mcclips(1), *mcpanel*(1)

mvcp(5)

NAME

mvcp - Multiport Video Computer Protocol

PROTOCOL REVISION

Revised 01MAR2003

DESCRIPTION

The Media Server for Broadcast (MSB) includes a TCP/IP-based protocol processor enabling an application remote, network-based control over video codec and storage resources comprising an MSB infrastructure. The Multiport Video Control Protocol (MVCP) is text-based and simplifies interaction between a client controlling application and the MSB.

PROTOCOL OVERVIEW

MVCP comprises a simple request/response protocol over a standard stream socket connection. The protocol semantics substantially match the standard file transfer protocol FTP. MVCP requests consist of case-sensitive command strings two to four characters long with a varying number of space-delimited arguments. All commands consist of uppercase ASCII letters. Each command and optional argument list must terminate with a CR/LF (HEX 0D/0A) combination.

NOTE: Lower case MVCP command strings will not parse. The command arguments may consist of any case combination of ASCII characters. Unprintable ASCII character passed as arguments are parsed as spaces, but retain their ASCII values, rendering these clips unmanagable by the MSB. Users should restrict command argument values to the inclusive ASCII range HEX 20 (space) and HEX 7E (~). Arguments with spaces as values must enclosed with quotation marks (HEX 22).

Protocol response message text consists of a numeric result code and an attributed informational message terminated by a CR/LF. Success is indicated with a response code of the form 2xx. Unless otherwise stated, a command returns the result code 200 if successfully parsed. However, a valid command parse does not automatically imply the underlying MSB resource executes successfully. Refer to the section on *UNIT COMMAND MODES* for discussion of this behavior.

Like FTP, certain MVCP responses may also include a single response line or multiple response lines terminated by a single blank line (CR/LF). In this case, a result code of 202 is returned and a single response line follows. Result code 201 returns if (one or more lines terminated by a blank line) is subsequently generated.

All MVCP command arguments and responses are white space-delimited. Command arguments that contain spaces must be encapsulated by double quotation marks to enable consistent argument parse. In some cases, response arguments that do not contain spaces may be double-quoted, and the application should be prepared to handle these specifically.

NOTE: Future MVCP enhancements may alter command name, argument quantity,

or reply results. When new arguments are added to a command, omission of the new arguments will result in the behavior associated with the original command; hence, existing applications will continue to work in a compatible manner. See the VERS command for information regarding protocol versioning, which aids the conditionalization of applications that track MVCP evolution.

UNIT MANAGEMENT

A single MVCP session can control multiple MSB logical units. Each unit consists of a logical VTR transport capable of loading, cueing, and playing a clip, and permits numerous other content operations, as defined and supported by MVCP. MVCP commands addressing a specific unit include the unit name as the first argument, while global commands, commands that affect MSB state or device categories, do not include a unit name.

For example:

```
 CINF clip1     (global: retrieve clip information)
 202 OK
 clip1 movie/stream/mpeg/mpeg2/sgi ... (additional metadata not shown)

 LOAD U1 clip1  (unit: load clip into unit U1)
 202 OK
 clip1 movie/stream/mpeg/mpeg2/sgi ... (additional metadata not shown)
```

When the unit name is omitted or specified as '*' with a unit command, the current unit is used. The unit name can be omitted only if the unit command has no required arguments.

Units can either be created (using the **UADD** command) or opened (using the **UOPN** command). When a unit is opened, it must have previously been created by another MVCP session (which could be another MVCP session or an external control processor such as a station automation system or on-line editor). Exercise caution when sharing control of a unit between two MVCP sessions. Interfering with a unit owned by an MSB external protocol processor (e.g., Sony, Harris, Odetics) may lead to unpredictable behavior.

COMMAND SETS

Two disjoint command subsets comprise MVCP. One subset includes global commands, those commands which address MSB state or device categories and do not address specific unit configuration or state. The other subset comprises unit commands. Certain commands affect only the current MVCP session invoking them, while others impact global MSB state.

GLOBAL COMMANDS: ACCESS CONTROL

USER *username*

The **USER** command sets the *username* for access control purposes. If MVCP access is restricted by the **mvcp.allow** or **mvcp.deny** files, the **USER** and **PASS** commands must be issued before access to any other MVCP commands is granted.

The **USER** command applies only to the current MVCP session.

The files /usr/vtr/config/mvcp.allow or /usr/vtr/config/mvcp.deny must be specified to enable and disable individual user access to MSB via MVCP. A comma-separated list of user names must be given in each file. The user names must match the entries held in /etc/passwd to affect comparison during session initiation.

If successful, the USER command returns 331 Password required for user.

PASS *password*
The **PASS** command verifies *password* for access control purposes and must follow the corresponding **USER** command.

The **PASS** command applies only to the current MVCP session. The /etc/passwd value for the user's password must contain a valid, non-null entry to affect comparison.

If successful, the PASS command returns 200 OK.

NOTE: The access controls furnished with USER and PASS commands are not fully implemented and usage should be avoided.

VERS
The **VERS** (Version) command reports the MVCP version supported by MSB. This may change with each release, signifying that command arguments have changed or that command semantics are different.

If successful, the VERS command returns 202 OK, followed by the MVCP version label. This command is currently unimplemented.

GLOBAL COMMANDS: PORTS

PLS The **PLS** (List Ports) command returns a list of supported media ports. If successful, a response code 201 returns and one response line for each port is returned. The format of each line is:

name mode "*description*" *type port-physical-name*

where:

name identifies the port name as known to MSB.

mode identifies the port's input/output mode (**IN**, **OUT**, or **BOTH**).

description provides descriptive text about the port.

type identifies the port type. Currently MSB supports port types of **NET** (network), **VID** (video), **DECK** (deck control), or **DISK** (disk storage).

port-physical-name defines the media ports physical name used for unit creation and management.

For example:

```
PLS
201 OK
MPEG422_6 OUT "SGI PCI-VIDAUD-MSB-B" VID "WHITE HOUSE OVAL OFFICE"
```

```
  MPEG422_7 OUT "SGI PCI-VIDAUD-MSB-B" VID "MAIN CASINO, ROULETTE TABLE 2"
  MPEG422_12 IN "SGI PCI-VIDAUD-MSB-B" VID "COUNT ROOM"
  MPEG422_13 OUT "SGI PCI-VIDAUD-MSB-B" VID "MAIN AND 1ST TRAFFIC LIGHT EAST
BOUND"
```

GLOBAL COMMANDS: UNITS

UADD *media-port-name storage-port-name port-sharing-mode* [*owner-info*]

The **UADD** command creates a new unit in the current MVCP session. The name of the media port accessed by the unit is specified by *media-port-name*. The storage port accessed by the unit is specified by *storage-port-name* (use '*' to specify the default storage system).

Certain media ports have no attributed storage port, such as those assigned to deck control. Specify 'null' for the storage port to create a unit which does not have a storage port.

The *port-sharing-mode* specifies how access is shared between multiple units accessing the same media port and takes one of the following values: **EXCL**, **SHAR**, or **CONC**.

Exclusive access (**EXCL**) asserts that only one unit may exist at any instance, precluding creation of additional units through contemporaneous MVCP sessions. UADD will fail if a pre-existing unit was created with EXCL and currently active within a sibling MVCP session.

Shared access (**SHAR**) allows multiple unit creation on the same port. However, only one unit may actually use the hardware at any given time. Shared access is only available on media ports which support it (in their interface module).

Concurrent access (**CONC**) allows multiple units to simultaneously use the media port. This mode is used most commonly on multiplexed networking ports.

The *owner-info* is optional and establishes an ownership label for the new unit. The owner info is output by certain unit status commands (UINF, UGIN, and ULS).

If the unit is successfully created, a response code 202 returns and a single response line containing the newly created unit name is output.

For example:

```
  UADD MFCODEC_6 * EXCL *
  202 OK
  U1
```

UGIN *unit-name*

The **UGIN** (Get Unit Info) command returns the owner and port information for the unit specified by *unit-name*. The specified unit does not need to be opened by the current MVCP session. **UINF** returns the identical information for an opened unit.

If the unit exists, a response code 202 returns and a single response line is returned with this format:

owner port-name port-mode port-physical-name

where:

owner identifies the creating MVCP session name for the unit.

port-name identifies the media port label controlled by the unit.

port-mode identifies the input/output mode the unit supports. The possible values are **IN**, **OUT**, and **BOTH**.

port-physical-name identifies the media port physical name controlled by the unit.

For example:

```
UGIN U1
202 OK
* MPEG422_6 OUT MFCODEC_6
```

ULS The **ULS** (List Units) command returns a list of previously active or existing MSB units. The current state of each unit is output.

If the **ULS** command is successful, a response code 201 returns and one response line for each unit is output followed by a terminating blank line. The display format for unit state is:

name owner port mode clip status function location speed rate command-id

where:

name identifies the unit.

owner identifies the MVCP session name that created the unit.

port identifies the unit's media port label.

mode identifies the media port mode (**IN**, **OUT**, or **BOTH**).

clip identifies the loaded clip name ("*****" if no clip is currently loaded).

status states the unit's current function status (**BUSY** - OP IN-PROGRESS, **RUN** - RUNNING, **DONE** - COMPLETE, **ERR** - ERROR).

function identifies the current function: **IDLE**, **LOAD**, **UNLD** (Unload), **CUE** (Cue for playout), **CUER** (Cue for record), **PLAY**, **STEP**, **SHTL** (Shuttle), **REC** (Record), **PAUS** (Pause), **STON** (StandbyOn), **STOP**, **FF** (Fast Forward), **REW** (Rewind).

location identifies the current clip location in hours:minutes:seconds:frames format. In drop-frame mode, a period (.) replaces the last colon in the timecode.

speed specifies the current playback speed (1000 = normal 1x speed).

rate specifies the frame rate (in frames/sec) corresponding to the unit location. 525-line drop-frame timing is specified in its approximate form, 29.97 (525 non-drop-frame is 30). 625-line is represented as 25 frames per second.

command-id identifies the unit's MVCP command in-process.

For example:

```
ULS
201 OK
U1 * MPEG422_6 OUT * DONE IDLE * 0 * 0
```

MON [*unit-name* ...] [*/ event-type* ...]

The **MON** (Monitor) command places the current MVCP session into event monitoring mode. In this mode, a single response line is returned whenever a monitored event occurs. If one or more units are specified, unit events for only the specified units are returned; Otherwise, events for all active units are returned.

A list of one or more event types may be specified (preceded by a single forward slash). The event types are: **UADD** (unit added), **URM** (unit closed), **UCHG** (unit state change), **ULOC** (unit location change), **UCTL** (unit control change), **UERR** (unit error), **CADD** (clip added), **CRM** (clip removed), **CCHG** (clip media modified), **CMV** (clip moved), **CEDP** (change in clip in/out points) and **CCHP** (clip attribute protection change). Only the specified types of events are returned.

If no event-type is specified, **MON** returns **UCHG**, **URM**, **UERR**, and **UCTL** events by default, and if no units are specified, **UADD**-related events are returned by default.

Event monitoring terminates by closing the MVCP session.

The event response line returned by **MON** consists of one of the following.

In response to **UADD** from an MVCP session, **MON** produces:

UADD *unit-name owner_info media_port_name port_sharing_mode physical_port_name*

UCHG *unit-name clip status function location speed rate command-id*

where:

unit-name is the name of the unit.

clip identifies the loaded clip name ("*****" if no clip is loaded).

status states the unit's current function status (**BUSY** - INITIALIZATION, **RUN** - RUNNING, **DONE** - COMPLETE, **ERR** - ERROR).

function states the unit's current function: **IDLE**, **LOAD**, **UNLD** (Unload), **CUE**, **CUER** (Cue for record), **PLAY**, **STEP**, **SHTL** (Shuttle), **REC** (Record), **PAUS** (Pause), **STON** (StandbyOn), **STOP**, **FF** (Fast Forward),

REW (Rewind).

location is the current clip location in hours:minutes:seconds:frames format. A period (.) replaces the final colon (:) in drop-frame mode.

speed identifies the current playback speed (1000 = normal 1x speed).

rate is the frame rate (in frames/sec) corresponding to the unit location. 525-line drop-frame timing is specified in its approximate form, 29.97 (525 non-drop-frame is 30). 625-line is represented as 25 frames per second.

command-id identifies unit's in-process command.

In response to **UCLS** from an MVCP session, **MON** produces:

URM *unit-name*

unit-name identifies the unit name.

In response to **ULS** from an MVCP session, **MON** produces:

ULOC *unit-name location rate*

unit-name identifies the unit name.

location identifies the current clip location in hours:minutes:seconds:frames format. A period (.) replaces the final colon (:) in drop-frame mode.

rate identifies the timecode rate (in frames/sec) corresponding to the unit location. 525-line drop-frame timing is specified in its approximate form, 29.97 (525 non-drop-frame is 30).

In response to a **SSET** from an MVCP session, or in response to initial unit creation via **UADD**, **MON** produces:

UCTL *unit-name control-name* "*control-value*"

unit-name identifies the unit name.

control-name identifies the control name.

control-value identifies the new value assigned to *control-name*.

In the event of a unit error, **MON** produces:

UERR *unit-name error-code* "*error-message*"

unit-name identifies the unit name experiencing the error.

error-code identifies the error code number.

error-message identifes textual error message.

In response to **CADD**, **MON** produces:

CADD *clip format size resident-size start end in out rate time-of-last-modification clip-type*

clip identifies the clip name.

format identifies the clip format (mpeg2, dvcpro, vframe, etc).

size identifies clip content size in bytes.

resident-size identifies the cache-resident clip content size in bytes

start states the clip timecode of the first time.

end states the clip timecode of the last frame.

in identifies the clip current mark-in timecode.

out identifies the clip current mark-out timecode.

rate identifies the clip frame rate (in frames/sec). 525-line drop-frame timing is specified in its approximate form, 29.97 (525 non-drop-frame is 30). Drop-frame and non-drop-frame translation is not performed irrespective of FRAT setting.

time-of-last-modification specifies the clip modification time, the last time the clip content was modified and flushed to mass storage.

clip-type specifies either CL or LN. The value LN implies that clip creation was performed as a link to another clip. The value CL implies the clip is unassociated.

In respone to **CRM**, **MON** produces:

CRM *clip*

clip identifies the clip name.

In response to **CCHG**, **MON** produces:

CCHG *clip format size resident-size start end in out rate time-of-last-modification clip-type*

clip identifies the clip name.

format identifies the clip format.

size specifies clip content size in bytes.

resident-size specifies cache-resident clip content size in bytes.

start identifies the clip timecode of the first frame.

end identifies the clip timecode of the last frame.

in identifies the clip current mark-in point.

out identifies the clip current mark-out point.

rate identifies the clip frame rate (in frames/sec). 525-line drop-frame timing is specified in its approximate form, 29.97 (525 non-

drop-frame is 30). Drop-frame and non-drop-frame translation is not performed regardless of FRAT setting.

time-of-last-modification specifies the clip modification time, the last time the clip content was modified and flushed to mass storage.

clip-type specifies either CL or LN. The value LN implies that clip creation was performed as a link to another clip. The value CL implies the clip is unassociated.

In response to **CMV**, **MON** produces:

CMV *clip new-clip*
clip identifies original clip name; *new-clip* identifies new clip name.

In response to **CEDP**, **MON** produces:

CEDP *clip mark-in mark-out* [*tc-track*]

mark-in mark-out identify the new *clip* in-out points. The optional argument *tc-track* specifies the time code track used for the *mark-in* and *mark-out*:

CLIP is the control track, **VITC** is the vitc track,

LTC is fis the ltc track. If not specified, the default time code track will be used.

In response to **CCHP**, **MON** produces:

CCHP *clip* {*protection-type ...*}

The returned protections values identify the new protections for *clip*. The protection values are **ATTR** (Attribute Protect), **MV** (Rename Protect), **REC** (Record Protect), and **RM** (Delete Protect).

UOPN *unit-name*
The **UOPN** (Unit Open) command permits the current MVCP session to control the unit.

If the unit opens successfully, a response code 202 and separate response line containing the unit name returns.

GLOBAL COMMANDS: CLIP EDITING

CBLD *clip new-clip* [*start*]
The **CBLD** command (Build Clip) creates *new-clip* by copying segments specified by *clip*. No association between *new-clip* and the original clip exists when the copy completes.

The source *clip* must possess a segmented format (i.e. movie/vclip), and be complete a virtual clips. Fetal vclips cannot be processed with **CBLD**. All segments in the source *clip* must be contiguous.

If *start* is given, the timecode of the first frame of *new-clip* is set to *start*; Otherwise, the first frame of *new-clip* is assigned that of *clip*.

A response code of 200 indicates success.

NOTE: CBLD requires the input *clip* to consist exclusively of I-frame content, since editing streams based on groups of picture is not supported. Building clips derived from different bitrates and chroma formats is discouraged.

CCLS [*clip*]

The **CCLS** command (Clip Close) ends clip editing operations for the open clip *clip*, or for the last clip opened or created if *clip* is not specified. The clip must have been previously opened or created via **COPN** or **CMK** during the current MVCP session. Any changes made to the clip since it was opened or created are committed to the clip cache. Subsequent clip editing operations for the clip must be preceded by another **COPN** command.

A response code of 200 indicates success.

CFCL *clip track-mask in out*

The **CFCL** (Clip Clear Frames) command clears (erases) frames of the *clip* from the frame specified by *in* (inclusive) to the frame specified by *out* (exclusive). The video and audio is removed, but the intermediate frames remain as filler. This contrasts with **CFRM** which removes the frames, closing the clip gap. This command is only applicable to fixed format clip content (DV, DVCPRO, or DIF, for instance). Frame clearing via CFCL is not supported by the PCI-VIDAUD-MSB-B codec.

This command is analogous to the **FCLR** command, except that it does not take place in the context of a unit. The clip must have previously been opened or created via **COPN** or **CMK** during the current MVCP session.

track-mask must be specified as "*".

A response code of 200 indicates success.

CFNW *clip track-mask in out*

The **CFNW** command (Clip Insert New Frames) inserts empty (black) frames into *clip* in the interval [*in*, *out*), between the original frames *in* and *in*+1. This command is only applicable to fixed format clip content (VC, DVCPRO, or DIF for instance).

This command is analogous to the **FNEW** command, except that it does not take place in the context of a unit. The clip must have previously been opened or created via **COPN** or **CMK** during the current MVCP session.

track-mask must be specified as "*".

A response code of 200 indicates success.

CFRM *clip track-mask in out*

The **CFRM** (Clip Remove Frames) command removes frames from *clip* beginning with frame *in* (inclusive) to the frame *out* (exclusive). This command removes the intermediate frames altogether, joining the

frames in the timeline to close the gap. This contrasts with **CFCL** which erases the video and audio but does not remove the intermediate frames. This command is only applicable to fixed format clip content (DV, DVCPRO, or DIF, for instance).

This command is analogous to the **FRM** command, except that it does not take place in the context of a unit. The clip must have previously been opened or created via **COPN** or **CMK** during the current MVCP session.

track-mask must be specified as "*".

A response code of 200 indicates success.

CMK *clip format-name*
The **CMK** command (Clip Make) creates *clip* with format *format-name*. *format-name* must specify the segmented format movie/vclip; non-segmented formats must be created within the context of a unit using the **LOAD** command.

If succesful, response code 200 is returned, and the new clip possesses a "fetal" state. In this state, the clip will not appear in the clip cache, although it can be deleted using **CRM**. In order for the clip to be visible in the clip cache and usable in the system, an initial segment must be added to it via the **CUPS** command. At that point, the new clip file will be written to the clip cache and become usable within the system.

The new clip will remain open for subsequent clip editing operations (see **COPN**) within the MVCP session duration or until explicitly closed with the **CCLS** command.

COPN *clip*
The **COPN** (Clip Open) command opens the existing *clip* for use in subsequent clip editing operations, as supported and specified with the commands in this section. The *clip* remains open for the duration of the current MVCP session, or until explicitly closed via **CCLS**.

CSAV [*clip*]
The **CSAV** command (Clip Save) causes the *clip*, or the most-recently opened or created clip, to have changes flushed to the clip cache. In general, clip editing operations are not committed to the clip cache until either a **CSAV**, **CCLS** or the end of the current MVCP session. However, real-time MSB resource and processing constraints preclude immediate guarantee of this behavior.

The clip must have been previously opened by a **COPN** or **CMK** command in the current MVCP session.

A response code of 200 indicates success.

CSCL *clip track-mask timecode*
The **CSCL** command (Clip Clear Segment) removes the segment beginning at the frame specified by *timecode* from *clip*. *track-mask* must be "*". The video and audio contained by the indicated segment is removed, but the frames remain (a gap is created in the clip). This contrasts with **CSRM**, which joins the interval. This command is only

applicable to fixed format clip content (DV, DVCPRO, or DIF, for instance). Segment clearing via CSCL is not supported by the PCI-VIDAUD-MSB-B codec.

The clip must have been previously opened by a **COPN** or **CMK** command in the current MVCP session.

A response code of 200 indicates success.

CSLS [*clip* [*track mask* [*in* [*out*]]]]

The **CSLS** command (Clip List Segments) lists the various segments that comprise the *clip*, or the most recently opened or created clip if *clip* is not specified. For a clip of segmented format (such as movie/vclip), this results in zero (0) or more response lines, depending upon the number of segments in the clip. For a non-segmented format (such as movie/vframe or movie/dif) it always results in a single segment response line, equal to the entire clip (similar to **CINF**).

If *in* is specified, only segments between *in* and the end are returned. If *out* is also specified, only segments between *in* (inclusive) and *out* (exclusive) are returned. If specified, *track-mask* must be specified as "*".

The clip must have been previously opened by a **COPN** or **CMK** command in the current MVCP session.

If succesful, a response code of 201 and the following per-segment information will be returned in the following format:

trk in out clip src-trk src-clip src-in src-out

trk is universally "*".

in identifies the segment in-point in the clip's timeline.

out identifies the segement out-point in the clip's timeline.

clip identifies the clip name in the clip-cache corresponding to this segment. Multiple segments may share the same value for *clip* if they are associated with the same source clip. The associated clip will link (see **CLN**) the original source clip, or an actual clip, if the segment was created via a **REC** command in the unit context. This clip is managed automatically by the system and should generally be ignored.

src-trk is universally "*".

src-in identifies the segment in-point of the original source clip.

src-out identifies the segment out-point of the original source clip.

src-clip identifies original source clip name from which this segment was taken, or "*" if the segment was created as a result of a **REC** command in the unit context.

CSRM *clip track-mask timecode*

The **CSRM** command (Clip Segment Remove) removes the segment beginning

at the frame specified by *timecode* from the clip specified by *clip*. *track-mask* must be "*". This command removes the frames altogether, moving the frames after the removed frames down in the timeline to close the gap. This contrasts with **CSCL** which simply erases the video and audio associated with the frames but does not remove the frames themselves. This command is only applicable to movie/vclip format.

The clip must have previously been opened or created via **COPN** or **CMK** during the current MVCP session.

A response code of 200 indicates success.

CUPS *clip src-op dest-op trk-mask in out src-clip src-trk-mask src-in src-out*

The **CUPS** command (Clip Update Segment) adds segments from an existing source clip to an existing destination clip. The destination clip must have a segmented format (i.e. movie/vclip), and must have previously been opened or created in the current MVCP session with **COPN** or **CMK**. The source clip need not be open.

A segmented virtual clip (vclip) links segments from source content clip files into a structure for playout or record. The links specify in and out timecodes pointing to source content clips. **CUPS** populates fetal vclips with video segments extracted from clip content files.

The new segment can be either inserted into the destination clip's timeline or can overwrite audio/video in that timeline (see *dest-op* below).

A homogenous vclip structure is enforced. Mixing clip formats (DV, DVCPRO, MPEG2, or DIF, for instance) is not permitted.

Streaming source clips (primarily MPEG) integrated into a segmented vclip via **CUPS** are ineligible for direct play out or record operations (see CBLD). Fixed frame formats such as DV or DVCPRO are eligible for segmented vclip playout or record operations.

The specified frames in the source clip can be removed, cleared, or left unchanged. In addition, if the source clip is itself segmented, the new segment in the destination clip can either reference the source clip itself, or the appropriate segments in the source clip.

In this later case, if the source clip is changed in the future, it will not affect the audio/video in the destination clip (see *src-op* below).

clip identifies the destination clip name, must possess a segmented format, and be open within the context of the current MVCP session. *src-op* specifies an operation performed against the source clip's specified frames.

The possible operations are:

FCP - Copy the appropriate segments from the source clip to the

destination clip. Both the source and destination must be segmented vclips (see CMK). If the source vclip changes, the destination vclip will not be altered. However, if the underlying source content clip changes, the destination vclip will change, as the copied segments directly reference the source content clip.

FLN - Link the entire source clip into the destination clip as a single segment. Unlike the **FCP** operation, if the source clip changes in the future, the destination clip is immediately affected. Also unlike **FCP**, the source clip is not required to have a segmented format. This operation is the only choice for an unsegmented source clip.

FRM - Remove the appropriate frames from the source clip after they have been copied to the destination clip. The source clip must have a segmented format. This operation implies copy semantics as opposed to link semantics as described in the **FLN** and **FCP** operations. Refer to **CFRM** for a full description of frame removal.

FCL - Clear the appropriate frames from the source clip after they have been copied to the destination clip. The source clip must have a segmented format. This choice implies copy semantics as opposed to link semantics as described in the **FLN** and **FCP** operations. Refer to **CFCL** for a full description of frame clearing. The PCI-VIDAUD-MSB-B card does not support frame clearing.

dest-op specifies whether the new segment should be inserted (**FINS**) into the destination clip or should overwrite (**FOVR**) any part of existing segments.

trk-mask must be specified as "*".

in identifies the new segment in-point for the destination clip's timeline.

out identifies the new segment out-point in the destination clip's timeline.

src-clip identifies the source clip name. It need not be open. The source clip audio and video format parameters must match the destination clip, unless the destination clip possesses a fetal as described in **CMK**.

src-trk-mask must be specified as "*".

src-in identifies the segment in-point for the source clip timeline. If specified as "*", the source clip in-point value is used.

src-out identifies the segment out-point for the source clip timeline. If specified as "*", the source clip out-point value is used.

If successful, the response code is 200.

GLOBAL COMMANDS: CLIP MANAGEMENT

CADD *clip* [*format*]

The **CADD** (Add Clip) command introduces new clip content to MSB inserted through an external mechanism to the clip cache. *clip* identifies the new clip name. *format* specifies the clip format. If *format* is not specified, the *clip clip.format* attribute is used if it exists, or MSB attempts to automatically discern clip format. See *attr*(1) for information on file attribute extensions.

The clip content must exist within a specific location addressable by the clip cache prior to issuing **CADD** (typcally /usr/vtr/clips). If the clip format requires an ancillary index (as is common for MPEG formats), the index file must also exist in the appropriate index directory (typically /usr/vtr/index) before issuing **CADD**. Refer to *vtrftp*(1) for a description of the mechanism preferred for external clip transfer to MSB.

CADD need not be issued to resolve a clip for use by other MSB commands (such as causing it to load). MSB automatically attempts to resolve and reference a clip when an MVCP session invokes a command that explicitly names it.

CADD adds a clip to MSB's clip cache such that it appears when CLS is invoked. CADD also generates the appropriate clip-add event messages for MVCP sessions that monitor the clip cache.

If a clip currently exists within the clip cache, CADD causes MSB to update the size and attributes if the content has changed in response to an external process.

CCHP *clip* {{+|-}*protection-type* ...}

The **CCHP** (Change Clip Protection) command adds to and/or removes protection from attributes of *clip* specified by the *protection-type* arguments.

The protection attribute values include **ATTR** (Attribute Protect), **MV** (Rename Protect), **REC** (Record Protect), and **RM** (Delete Protect).

When set, **ATTR** value precludes clip metadata modifications, such as edit in/out point modification or frame removal (CFRM) or clear operations (CFCL). When set, **MV** precludes the clip from rename via CMV. When set, **REC** attribute precludes the clip from record or append. When set, **RM** precludes clip deletion via CRM.

For example, the command "CCHP NA1001 +RM -ATTR" sets the protection on NA1001 such that the clip cannot be deleted but can have its attributes (such as its edit points) changed.

If successful, the response code is 200.

CCP *clip new-clip*

The **CCP** (Copy Clip) command creates by copying *clip* to new clip *new-clip*. After the copy is created, no association exists between *new-clip* and *clip*.

If successful, the response code is 200.

For example:

```
  CCP palmpeg0 newclip
  200 OK
```

NOTE: CCP does not force realign a clip as the copy proceeds. Certain clip formats, such as MSB vframe, require destination filesystems to match major and minor alignment parameters of the source.

CCST

The **CCST** (Clip Cache Status) command returns the current status of the MSB Clip Cache. The response code is 202 and a single response line is returned in the following format:

num-clips bytes-used bytes-avail bytes-avail-contiguous

where:

num-clips states the current count of cache-resident clips.

bytes-used states the total bytes used by the clip-cache.

bytes-avail states the available free bytes within the storage systems to contain the clip cache.

bytes-avail-contiguous states the largest available contiguous amount of free bytes in the storage system.

For example:

```
  CCST
  202 OK
  1 53673984 436716306432 436662632448
```

CEDP *clip mark-in mark-out*

The **CEDP** (Set Clip Edit Points) command sets the edit points *clip*. The *mark-in* and *mark-out* arguments are specified in HH:MM:SS:FF format. The new edit points will not be seen by any unit until the clip is loaded (or reloaded). If the clip is already loaded into a unit, the original edit points will apply.

Specifying '*' for *mark-in* or *mark-out* removes the respective edit point.

A response code 200 returns if the command succeeds.

CGP *clip*

The **CGP** (Get Clip Protection) command returns the current protection attribute values for *clip*.

If successful, the response code 202 returns, and a single response line is returned containing the protection attribute values currently enabled for the clip. The allowed attribute protection values are **ATTR** (Attribute Protect), **MV** (Rename Protect), **REC** (Record Protect), and **RM** (Delete Protect).

When set, **ATTR** value precludes clip metadata modifications, such as edit in/out point modification or frame removal (CFRM) or clear operations (CFCL). When set, **MV** precludes the clip from rename via

CMV. When set, **REC** attribute precludes the clip from record or append. When set, **RM** precludes clip deletion via CRM.

CIMG *clip timecode interleave filename* [*format*]

The **CIMG** (Create Clip Image) command extracts the image data associated with the frame specified by *timecode* of *clip* and writes the image to *filename*. This command is only applicable to fixed format clip content (DV, DVCPRO, or DIF, for instance).

interleave specifies how to construct the image from the two fields of the frame: F1 (odd field only), F2 (even field only), F1F2 (both fields interleave), F1F1 (odd field line-doubled), F2F2 (even field line-doubled).

format specifies the image format to use. Possible values are "rice", "rgb", "yuv", "jpeg", and "tiff". If *format* is not specified, the format is inferred from the filename extension (".rice", ".rgb", ".yuv", ".jpg", ".tiff").

A response code 200 returns if the command succeeds.

CINF *clip*

The **CINF** (Clip Info) command returns the metadata attributes of *clip*. If the *clip* is currently clip-cache resident, the response code is 202 and a single response line is returned in the following format:

clip format size resident-size start end in out rate time-of-last-modification clip-type

clip identifies the clip name.

format identifies the clip format.

size identifies clip content size in bytes.

resident-size identifies the cache-resident clip content size in bytes

start states the clip timecode of the first time.

end states the clip timecode of the last frame.

in identifies the clip current mark-in timecode.

out identifies the clip current mark-out timecode.

rate specifies the frame rate (in frames/sec) corresponding to the unit location. 525-line drop-frame timing is specified in its approximate form, 29.97 (525 non-drop-frame is 30). Translation between drop-frame and non-drop-frame is not performed regardless of FRAT setting.

time-of-last-modification states clip modification time and disk update, and possesses the format yyyymmddThhmmss.microsecsZ. For example, a value of 19990316T020942.836820Z means the clip was last modified 16th March, 1999 at two hours, nine minutes, forty two seconds and 836820 microseconds past midnight GMT.

clip-type specifies either CL or LN. The value LN implies that clip creation was performed as a link to another clip. The value CL implies the clip is unassociated.

CLN *clip new-clip*
The **CLN** (Link Clip) command links *new-clip* with the clip content of *clip*. The clip attributes (such as edit points) of the new clip may be set independently of the original clip. This command applies to all clip formats, fixed (DV, DVCPRO, DIF) and variable (MPEG2).

If the original clip is deleted, the new clip retains the the original clip content until the new clip is also deleted.

This command is useful for creating clips which refer to segments of other clips.

If successful, the response code is 200.

CLS *pattern*
The **CLS** (List Clips) command returns a list of all clips in the MSB **clip cache**. The response code is 201 and one response line is returned for each cache-resident clip. An optional argument *pattern* can be supplied to limit the returned list to only clips matching the pattern. See **sh(1)** under "Filename Generation" for the *pattern* syntax restrictions. The response line format is:

clip format size resident-size start end in out rate time-of-last-modification clip-type

clip identifies the clip name.

format identifies clip format.

size identifies clip content size in bytes.

resident-size identifies the cache-resident clip content size in bytes

start states the clip timecode of the first time.

end states the clip timecode of the last frame.

in identifies the clip current mark-in timecode.

out identifies the clip current mark-out timecode.

rate specifies the frame rate (in frames/sec) corresponding to the unit location. 525-line drop-frame timing is specified in its approximate form, 29.97 (525 non-drop-frame is 30). Translation between drop-frame and non-drop-frame is not performed regardless of FRAT setting.

time-of-last-modification states clip modification time and disk update, and possesses the format yyyymmddThhmmss.microsecsZ. For example, a value of 19990316T020942.836820Z means the clip was last modified 16th March, 1999 at two hours, nine minutes, forty two seconds and 836820 microseconds past midnight GMT.

clip-type specifies either CL or LN. The value LN implies that clip creation was performed as a link to another clip. The value CL implies the clip is unassociated.

CLSA

The **CLSA** (List Added Clips) command lists clips added to the clip cache since the last time the **CLSA** executed (or since the **CMON** command which created this clip monitor was executed).

The **CMON** (Clip Monitor) command must be issued to initiate clip cache monitoring before CLSA is entered by the current MVCP session.

If successful, the response code is 201 and one response line is returned for each newly added clip which contains the clip's name. A blank line terminates the list of clips.

CLSL *clip*

The **CLSL** (List Clips With Same Base) command lists all clips (including itself) in the clip cache with the same base *clip*. It can thus be used to find all links to a given clip or the base clip for a given link.

If the command is successful, the response code is 201 and for each clip found, one response line is returned in the following format:

clip-name clip-type

clip-type contains LN (a clip link) or CL (an unassocated clip).

CLSR

The **CLSR** (List Removed Clips) command lists clips removed from the clip cache since the last time the **CLSR** command was issued (or since the **CMON** command which created this clip monitor was executed).

The **CMON** (Clip Monitor) command must be issued to initiate monitoring of the clip cache by the current MVCP session.

If successful, the response code is 201 and one response line is returned for each removed clip which contains the clip's name. A blank line terminates the list of clips.

CMIN

The **CMIN** (Clip Monitor Info) command returns the number of clips added to and/or removed from the clip cache as returned by the **CLSA** and **CLSR** commands respectively.

The response code is 200, and a single response line is returned in the following format:

num-clips-added num-clips-removed

where:

num-clips-added identifies the number of added clips.

num-clips-removed identifies the number of removed clips.

CMON

The **CMON** (Clip Monitor) command initiates clip cache monitoring. The **CLSA** and **CLSR** commands are used to retrieve, respectively, the clips that have been added or removed from the clip cache.

If the **CMON** command is issued for a second or subsequent time, the current list of added and removed clips is discarded and monitoring is initialized again.

The response code is 200.

CMV *clip new-clip*
The **CMV** (Move Clip) command renames *clip* to *new-clip*.

If successful, the response code is 200.

NOTE: **CMV** cannot be used to move clips between file systems; use **CCP** instead.

CRM *clip*
The **CRM** (Delete Clip) command deletes *clip*. If the clip is currently loaded by a unit, actual deletion is deferred until the clip unloads.

If the command is successful, the response code is 200.

For example:

```
CMV newclip oldclip
200 OK
```

CRMA
The **CRMA** (Delete All Clips) command deletes all clips currently residing in the clip cache.

If the command is successful, the response code is 200.

For example:

```
CRMA
200 OK
CLS
201 OK
```

CINT *clip*
The **CINT** (Interrupt Clip Operation) command interrupts the clip operation that is updating or creating *clip*.

Since clip operations are synchronous, a separate MVCP session must issue the **CINT** command.

If there is no clip operation associated with *clip* or the operation completed, the response code is 505, "Unable to interrupt clip operation".

If the command succeeds, the response code is 200.

GLOBAL COMMANDS: SYSTEM CONTROLS

SGET *subsystem-name control-name-pattern* ...

The **SGET** (System Get Control) command retrieves system control values for *subsystem-pattern*. The possible values for *control-name-pattern* are subsystem-dependent and are described in msb-*controls*(5). If the subsystem or control patterns contains wildcards, the values of all controls that match the specified patterns will be returned. The allowable *subsystem-pattern* values are main, clipmirror, and fs.

If successful, response code is 201, and one response line is returned for each matching control in the following format:

subsystem-name control-name "*control-value*"

For example:

```
SGET clipmirror *
201 OK
clipmirror vtr.clipmirror.local_server.hostname ""
clipmirror vtr.clipmirror.max_threads "20"
clipmirror vtr.clipmirror.primary_server.hostname ""
clipmirror vtr.clipmirror.reconnect_interval "30"
```

SSET *subsystem-name control1-name control1-value* ...

The **SSET** (System Set Control) command sets system controls for the subsystem specified by *subsystem-name*. For each control, a name and a value must be provided. At least one control name/value pair must be specified. If the control value contains any spaces or tabs, the value must be enclosed in double quotes. The allowable *subsystem-name* values are main and clipmirror.

The possible values for *control-name* and *control-value* are subsystem-dependent and in msb-*controls*(5).

For example:

```
SSET clipmirror vtr.clipmirror.max_threads 24
200 OK
SGET clipmirror *thread*
201 OK
clipmirror vtr.clipmirror.max_threads "24"
```

GLOBAL COMMANDS: STATISTICS

STLS [*component-pattern* [*statistic-pattern*]]

The **STLS** (List Statistics) command lists the component name, statistic name, and current value of each statistical value matching the specified patterns.

If the command is successful, the response code is 201, and a response line is returned for each matching statistical value in the following format:

component-name statistic-name value ...

where:

component name identifies the component instance name generating the statistic.

statistic name identifies the statistical value name.

value identifies the integer or floating-point statistical value. The value of certain types of statistics (e.g., histogram) may include more than one number.

STST [*component-pattern* [*statistic-pattern*]]
The **STST** (Statistics Statistics) command calculates various statistics over all of the statistical values matching the specified patterns.

If the command is successful, the response code is 202, and a single response line is returned in the following format:

values samples min max sum mean stddev

where:

values identifies the number of statistical values matching the pattern.

samples identifies the total number of samples collected.

min identifies the minimum value.

max identifies the maximum value.

sum identifies the sum of the values.

mean identifies the mean of the values.

stddev identifies the standard deviation of the values.

STZ [*component-pattern* [*statistic-pattern*]]
The **STZ** (Statistics Reset) command resets the values of all the statistics matching the specified patterns.

GLOBAL COMMANDS: OTHER

BYE The **BYE** command terminates the current MVCP session and disconnects from MSB.

ERR The **ERR** command returns the code and description for the last global error that occurred for this MVCP session. Errors that occur on units are retrieved using the **UERR** command. This command returns errors for all Clip Management commands and other commands which do not pertain to a specific unit.

The response code is 202, and a single response line is returned in the following format:

code "*description*"

where:

code identifies the error code.

description identifies the error description. If no error occured

from the last command a "*" is used for the message text.

For example:

```
ERR
202 OK
0 "*"
```

FRAT *frame-rate*
The **FRAT** (Frame Rate) command sets the frame rate used in translating timecodes for command timing and for unit operations. The *frame-rate* is specified as frames per second. Supported values are 25, 29.97, and 30.

The frame rate is set initially according to the value of the system control vtr.main.timing_standard.

The **FRAT** command applies only to the current MVCP session.

GTOD
The **GTOD** (Get Time-of-Day) command returns the current MSB system time. The response code is 202, and a single response line is returned with three forms of the current time (time code, ISO 8601, and Unadjusted System Time):

hh:*mm*:*ss*:*ff yyyymmddThhmmss.ssssssZ UST*

SORD *order-type*
The **SORD** (Set Sort Order) command sets the type of ordering used when lists are returned. The possible values of *order-type* are **NAME**, which sorts lists by clip name, and **TIME**, which sorts lists by creation time.

STOD *time*
The **STOD** (Set Time-of_Day) command sets the MSB system time as specified by *time* which is specified either as a time code (hh:mm:ss:ff) or in the ISO 8601-compatible format (yyyymmddThhmmss.ssssssZ).

UNIT COMMANDS

The unit commands all have as their first argument the name of unit to which the command is to be applied.

CUE [*unit-name* [*in* [*out* [*direction* [*passes* [*tc-track*]]]]]
The **CUE** (Cue For Play) command cues for playback the clip currently loaded by *unit-name*. If *in* is specified, the clip is cued at the specified frame. If *in* is missing or specified as '*', the mark-in point stored with the clip is used, or if mark-in is not set, the first recorded frame of the clip is used.

If *out* is specified, the clip is cued with the specified out point, meaning playback will terminate at the specified frame. If *out* is missing or specified as '*', the mark-out point stored with the clip is used, or if mark-out is not set, no out point is used.

If *out* is specified, *in* must be specified.

If either *in* or *out* is specified and not '*', the other may be specified as a duration by adding a '+' prefix character. For example, "1:00:01:00 1:00:06:00", "1:00:01:00 +5:00", and "+5:00 1:00:06:00" imply an identical edit range.

The optional *direction* argument specifies the playback direction: **FWD** is forward, **BWD** is backward, **F/B** is forward followed by backward, **B/F** is backward followed by forward. The default direction is forward (FWD).

The optional *passes* argument specifies how many passes through the clip are made. The default is 1 pass. Specify passes as 0 to repeat indefinitely (use the STOP command to terminate playout).

If *passes* is specified as -1, the unit is cued in free-range mode (for media devices that support free-range cueing). In free-range mode, the *in* point is used only as the initial location but does not define the lower limit of playback. The lower limit is defined by the vtr.media.clip.limit.start control, if set, or the start of the clip if the clip limit control is not set.

In free-range mode, the upper limit of playback is defined by the specified out-point. If the out-point is not set, the upper limit is defined by the vtr.media.clip.limit.end control, if set, or the end of the clip if the clip limit control is not set.

The optional *tc-track* argument specifies the time code track from which the edit points *in* and *out* are selected: **CLIP** is the control track, **VITC** is the vitc track, **LTC** is the ltc track. If not specified, the default time code track will be used.

If the command succeeds, the response code is 200.

CUER [*unit-name* [*in* [*out*]]]

The **CUER** (Cue For Record) command cues for recording the clip currently loaded by *unit-name*. If *in* is specified, the clip is cued at the specified frame. If *in* is missing or specified as '*', the unit cues to the mark-in point stored with the clip or if mark-in is not set.

If *out* is specified, the clip is cued with that out-point, meaning recording will terminate at the specified frame. If *out* is missing or specified as '*', recording will continue until the unit is stopped. If *out* is specified, *in* must also be specified.

If either *in* or *out* is specified and not '*', the other may be specified as a duration by adding a '+' prefix character. For example "1:00:01:00 1:00:06:00", "1:00:01:00 +5:00", and "+5:00 1:00:06:00" imply an identical edit range.

If the command succeeds, the response code is 200.

EDIT [*unit-name* [*in* [*out*]]]

The **EDIT** (Edit) command performs an auto-edit for the edit range *in* *out*. The unit prerolls for the duration specified by the setting of the vtr.edit.preroll control and postrolls for the duration specified

by the vtr.edit.postroll control. This command is applicable to tape-based decks.

If the command succeeds, the response code is 200.

FCLR *unit-name in out*
The **FCLR** (Clear Frames) command clears (erases) the frames of the clip loaded into *unit-name* from the frame specified by *in* (inclusive) to the frame specified by *out* (exclusive). The video and audio associated with the cleared frames is removed, but the frames themselves remain. This contrasts with **FRM** which removes the frames, closing the intermediate clip gap.

The clip must be loaded for input (**IN**) or input/output (**BOTH**).

If the command succeeds, the response code is 200.

FF [*unit-name*]
The **FF** (Fast Forward) command fast forwards the unit specified by *unit-name*. The fast forward speed is device-dependent.

If the command succeeds, the response code is 200.

FINS *unit-name source-in source-out dest-in*
The **FINS** (Insert Frames) command moves the frames in the range *source-in* to *source-out* of the clip loaded into *unit-name*, inserting them at the point specified by the timecode *dest-in*. The frames are removed from the original location, and the duration of the clip remains unchanged.

The clip must be loaded for input (**IN**) or input/output (**BOTH**).

If the command succeeds, the response code is 200.

FNEW *unit-name in out*
The **FNEW** (Insert New Frames) command inserts blank frames into the clip loaded into *unit-name* between the frame specified by *in* (inclusive) and *out* (exclusive). **FNEW** opens a clip gap, and the clip's duration increases.

The clip must be loaded for input (**IN**) or input/output (**BOTH**).

If the command succeeds, the response code is 200.

FOVR *unit-name source-in source-out dest-in*
The **FOVR** (Overwrite Frames) command moves the frames in the range *source-in* to *source-out* of the clip loaded into *unit-name*, overwriting the frames starting at the timecode specified by *dest-in*. The frames are removed from the original location. The duration of the clip will decrease, though if the source and target overlap, the duration will decrease by a fewer number of frames than are represented by the source range.

The clip must be loaded for input (**IN**) or input/output (**BOTH**).

If the command succeeds, the response code is 200.

FRM *unit-name in out*

The **FRM** (Remove Frames) command removes the frames of the clip loaded into *unit-name* from the frame specified by *in* (inclusive) to the frame specified by *out* (exclusive). This command removes the intermediate frames altogether, joining frames after the removed frames in the intermediate timeline. This contrasts with **FCLR** which erases the video and audio but does not remove the intermediate frames.

The clip must be loaded for input (**IN**) or input/output (**BOTH**).

If the command succeeds, the response code is 200.

GET *unit-name dev-type control-name-pattern ...*
The **GET** (Get Control) command retrieves the device control values for *unit-name*. The *dev-type* specifies whether the controls are queried in the media (**MED**) or storage (**STOR**) device assigned to the unit.

The possible values for *control-name-pattern* are device-dependent are described elsewhere. If the pattern contains wildcards, the values of all controls that match the specified pattern will be returned.

If successful, response code is 201, and one response line is returned for each matching control in the following format:

control-name "*control-value*"

For example:

```
UADD MFCODEC_6 * EXCL *
202 OK
U1
GET U1 MED *gop*
201 OK
vtr.media.video.input.compression.gop_on_scene_change "true"
vtr.media.video.input.compression.gop_size "1"
vtr.media.video.input.compression.gop_structure "I"
```

GOTO *unit-name timecode*
The **GOTO** (Goto) command jumps the unit transport *unit-name* to the location specified by *timecode*. The unit transport continues executing the currently active function when GOTO initiates and completes.

JOG *unit-name count*
The **JOG** command advances the queued clip content on *unit-name* ahead or behind *count* frames, simulating VTR jog transport. A negative *count* moves backwards, and a positive *count* moves forward.

LIMS *unit-name in* [*out* [*direction* [*passes*]]]
The **LIMS** (Set Edit Limits) command modifies the *unit-name* edit limits imposed on the current playback or record function. Edit parameter modification is device-dependent. In general, only the out-point may be changed to stop playback or recording at a specific point even if the unit was cued without an out-point.

LOAD *unit-name clip* [*load-mode* [*load-options*]]
The **LOAD** (Load Clip) command loads *clip* into *unit-name*. The *load-*

mode indicates whether the clip is being loaded for input (**IN**), output (**OUT**), or both input and output (**BOTH**). The default is output (**OUT**).

The *load-options* indicate special handling for the clip when loading it. Zero or more options may be specified. Possible values are:

CRTE Create the clip if it does not exist.

NOEX Fail if the clip exists.

EPHM Create an ephemeral clip (automatically deleted when the clip is unloaded).

If successful, the response code is 202 and a single response line is returned in the following format:

format size resident-size start end in out rate

clip identifies the clip name.

format identifies the clip format (mpeg2, dvcpro, vframe, etc).

size identifies clip content size in bytes.

resident-size identifies the cache-resident clip content size in bytes

start states the clip timecode of the first time.

end states the clip timecode of the last frame.

in identifies the clip current mark-in timecode.

out identifies the clip current mark-out timecode.

rate identifies the clip frame rate (in frames/sec). 525-line drop-frame timing is specified in its approximate form, 29.97 (525 non-drop-frame is 30). Drop-frame and non-drop-frame translation is not performed irrespective of FRAT setting. 625-line is represented as 25 frames per second.

PAUS [*unit-name*]

The **PAUS** (Pause) command pauses *unit-name*. The unit must be in a playback or record state. The **RSUM**, **PLAY**, **STEP**, **SHTL**, **FF**, or **REW** commands are used to resume playback. The **RSUM** or **REC** commands are used to resume recording.

If the unit is playing, the video output will be a still frame at the paused clip location.

If the command succeeds, the response code is 200.

PLAY [*unit-name* [*speed*]]

The **PLAY** command begins (or resumes) playback on *unit-name*. If *speed* is not specified, normal playback speed, 1000, is used. Some media ports may support other playback speeds.

Some media ports must be explicitly cued before starting playback. For those devices, **PLAY** cannot be used to resume playback after a **STOP** command is issued. The unit must be re-cued.

If the command succeeds, the response code is 200.

REC *unit-name*
The **REC** (Record) command begins recording on the unit specified by *unit-name*. **REC** can also be used to resume normal recording after a **PAUS** command.

Some media ports must be explicitly cued before record commences. For those devices, **REC** cannot resume recording after a **STOP** command is issued; the unit must be re-cued.

If the command succeeds, the response code is 200.

REVU *unit-name in out*
The **REVU** (Review) command performs an edit review on *unit-name* for the edit range *in* to *out*. The unit prerolls for the duration specified by the setting of the vtr.edit.preroll control and postrolls for the duration specified by the vtr.edit.postroll control.

If the command succeeds, the response code is 200.

REW *unit-name*
The **REW** (Rewind) command rewinds *unit-name*. The rewind speed is device-dependent and can be set with a control.

If the command succeeds, the response code is 200.

RHRS *unit-name in out*
The **RHRS** (Rehearse) command performs an auto-edit on *unit-name* for the edit range *in out*, but during the actual edit range switches the output to E-to-E mode rather than recording. The unit prerolls for the duration specified by the setting of the vtr.edit.preroll control and postrolls for the duration specified by the vtr.edit.postroll control.

If the command succeeds, the response code is 200.

RSUM [*unit-name*]
The **RSUM** (Resume) command resumes the playback or recording function that was paused by a PAUS command on *unit-name*.

If the command succeeds, the response code is 200.

SET *unit-name dev-type control1-name control1-value* ...
The **SET** (Set Control) command sets device controls for *unit-name*. The *dev-type* specifies whether the controls are to be set in the media (**MED**) or storage (**STOR**) device assigned to the unit.

For each control, a name and a value must be provided. At least one control name/value pair must be specified. If the control value contains any spaces or tabs, the value must be enclosed in double quotes.

The possible values for *control-name* and *control-value* are device-dependent and are described elsewhere.

If the command succeeds, the response code is 200.

For example:

```
UADD MFCODEC_6 * EXCL *
202 OK
U1
GET U1 MED *gop*
201 OK
vtr.media.video.input.compression.gop_on_scene_change "true"
vtr.media.video.input.compression.gop_size "1"
vtr.media.video.input.compression.gop_structure "I"

SET U1 MED vtr.media.video.input.compression.gop_on_scene_change
      false
200 OK

GET U1 MED vtr.media.video.input.compression.gop_on_scene_change
201 OK
vtr.media.video.input.compression.gop_on_scene_change "false"
```

SHTL *unit-name speed*

The **SHTL** (Shuttle) command shuttles *unit-name* at the speed *speed*, simulating VTR shuttle transport. A speed of 1000 is normal 1x speed. The speed scale is linear, so a speed of 100 is 1/10th normal speed and a speed of 10000 is 10x speed.

Positive speeds play forward. Negative speeds play backward. The actual shuttle speeds available are device-dependent.

If the command succeeds, the response code is 200.

STEP [*unit-name* [*frames*]]

The **STEP** (Step) command steps *unit-name* by a frames count *frames* (1 frame, if not specified). Positive numbers step forward. Negative numbers step backward. If the distance specified by *frames* takes the unit past the current edit limits, the unit will stop (or pause, if the unit is configured to pause instead).

Some media ports require that the unit already be in a playback state when **STEP** is issued.

SSPD *unit-name speed*

The **SSPD** (Set Speed) command sets the *unit-name* speed to *speed*. **SSPD** is used to change speeds without changing the current unit transport function (either **PLAY** or **SHTL**).

If the command succeeds, the response code is 200.

STON [*unit-name*]

The **STON** (Standby On) command halts playback but does not stop the *unit-name*. This releases the output port for use by another unit, but allows the unit to resume playback quickly when a new playback

command arrives.

STON is supported by only some media ports.

If the command succeeds, the response code is 200.

STOP [*unit-name*]
The **STOP** (Stop) command stops *unit-name* playout or record.

For some media ports, playback or recording cannot be resumed until the unit is re-cued with a **CUE** or **CUER** command, respectively.

If the command succeeds, the response code is 200.

UCLS [*unit-name*]
The **UCLS** (Unit Close) command closes *unit-name*. This MVCP session's access will be terminated, and if no other MVCP sessions have the unit open, the unit will be deleted.

If the command is successful, response code 200 is returned.

For example:

```
UCLS U1
200 OK
```

UERR [*unit-name*]
The **UERR** (Unit Error) command returns the code and description for the last error that occurred on the specified unit.

If the command is successful, the response code 202 is returned followed by a data line:

code "*description*"

where:

code identifies the last error code.

description identifies the error code text.

If the command succeeds, the response code is 200.

For example:

```
UCLS U1
200 OK
UOPN U1
510 Unable to open unit
UERR
403 Unit not open
```

UFLS [*unit-name*]
The **UFLS** (Flush Unit) command flushes the command queue for the *unit-name*.

If the command succeeds, the response code is 200.

UINF [*unit-name*]
The **UINF** (Get Unit Info) command returns the owner and port information for the unit specified by *unit-name*. The unit must be opened (or created) by the current MVCP session.

The response code is 202 and single response line is returned in the following format:

owner port-name port-mode port-physical-name

where:

owner identifies the MVCP session name responsible for unit creation.

port-name identifies the media port name controlled by the unit.

port-mode identifies the input/output mode supported by the unit. The possible values are **IN**, **OUT**, and **BOTH**.

port-physical-name identifies the media port physical name controlled by the unit.

UINT [*unit-name*]
The **UINT** (Unit Interrupt) command interrupts the command currently being processed by *unit-name*.

If the command succeeds, the response code is 200.

ULOC [*unit-name*]
The **ULOC** (Unit Location) command returns the current transport location for *unit-name*.

The response code is 202 and a single response line is returned in the following format:

clip-loc vitc ltc UTC-time UST-time rate

where:
clip-loc states the clip location, which corresponds to an approximate CTL (control track) timecode.

vitc states the frame VITC (vertical interval timecode).

ltc states the frame LTC (longitudinal timecode).

UTC-time states the unit UTC time at the specified location. This time can be used to account for application or network latency in time reporting. This time is reported in the ISO 8601-compatible format: yyyymmddThhmmss.ssssssZ.

UST-time states the UST (unadjusted system time) time at the specified location. This time can be used to account for application or network latency in time reporting.

rate states the timecode rate (in frames/sec) corresponding to the specified clip location. 525-line drop-frame timing is specified in its approximate form, 29.97 (525 non-drop-frame is 30). 625-line is represented as 25 frames per second.

UNLD [*unit-name*]
The **UNLD** (Unload Clip) command unloads the clip currently loaded by *unit-name*.

If successful, the response code is 200.

USTA [*unit-name*]
The **USTA** (Unit Status) command returns the status of the unit specified by *unit-name*. The unit my be opened (or created) by the current MVCP session.

If successful, the response code is 202, and a single response line is returned in the following format:

clip status function location speed rate command-id

clip identifies the loaded clip name ("*****" if no clip is loaded).

status states the unit's current function status (**BUSY** - OP IN-PROGRESS, **RUN** - RUNNING, **DONE** - COMPLETE, **ERR** - ERROR).

function identifies the current function: **IDLE**, **LOAD**, **UNLD** (Unload), **CUE**, **CUER** (Cue for record), **PLAY**, **STEP**, **SHTL** (Shuttle), **REC** (Record), **PAUS** (Pause), **STON** (StandbyOn), **STOP**, **FF** (Fast Forward), **REW** (Rewind).

location identifies the current clip location in hours:minutes:seconds:frames format. In drop-frame mode, a period (.) replaces the last colon (:).

speed specifies the current playback speed (1000 = normal 1x speed).

rate specifies the frame rate (in frames/sec) corresponding to the unit location. 525-line drop-frame timing is specified in its approximate form, 29.97 (525 non-drop-frame is 30). *command-id* identifies the unit's MVCP command in-process. 625-line is represented as 25 frames per second.

USYN [*unit-name*]
The **USYN** (Set Unit Synchronization) command sets the default command synchronization mode for the unit specified by *unit-name*. The mode is set as specified by the sync-mode qualifiers which precede the command name. See the section on *UNIT COMMAND SYNCHRONIZATION*.

Once the **USYN** command has been used to set the default synchronization mode, all unit commands which do not specify a sync mode will use this default.

UUWT *unit-name wait-unit-name sync-mode* [*wait-id*]
The **UUWT** (Unit-Unit Wait) command provides a mechanism for synchronizing command execution between two units. *unit-name* waits until *wait-unit-name* reaches execution status specified by *sync-mode* for *wait-id*.

For the possible values of *sync-mode*, see the section on *SYNCHRONIZATION* below. Both units must be opened by the current MVCP

session.

If *wait-id* is not specified, the unit waits for the command at the end of the target unit's command queue when the **UUWT** starts execution in the unit (not when the **UUWT** is issued to the unit).

The target command id can be obtained by using the /CID qualifier on the target unit command.

UWAT [*unit-name* [*sync-mode*]]
The **UWAT** (Unit Wait) command waits for the last command issued to *unit-name* according to the synchronization mode *sync-mode* or the default sync mode if *sync-mode* argument is not provided.

For the possible values of *sync-mode*, see the section on *SYNCHRONIZATION* below.

UNIT COMMAND MODES

The MSB units execute asynchronously with the MVCP session controlling them. How the MVCP control processor interacts with the units under session control and issuing MVCP commands is determined by command synchronization processing.

By default, when a command is issued to a unit, the unit's command queue is immediately flushed of any commands which are waiting to be executed by the unit, and the command preempts the previous command that was issued to the unit, even if that command has not completed execution.

Also by default, the MVCP session returns a response to the client as soon as the command queues for the unit. The generated reply in response to a command means that the command was successfully queued, not that the command completed successfully or started execution.

SYNCHRONIZATION

The synchronization mode determines when the MVCP session replies to the MVCP client. The available synchronization modes are: **ASYN** (async), **SYNI** (sync-init), **SYNR** (sync-run), and **SYNC** (sync-complete). These modes may be specified with USYN, or can prefix an MVCP command sequence.

The async (**ASYN**) mode implies the MVCP session receives a reply when the command queues, but immediate command initiation is not guaranteed, and may be deferred.

Sync-init (**SYNI**) mode delays response until the unit reaches command initialization state. This does not mean that the unit has initiated command execution, but that the command may block on a shared resource (such as the media port), and will wait for the resource to become available before continuing.

Sync-run (**SYNR**) mode delays response until the unit reaches the command initiated state. This means that the unit contemporaneously processes the command. For instance, the running state for a **PLAY** command means that video playout is currently active.

Sync-complete (**SYNC**) mode delays response until the unit reaches a

finished, terminal state. This means that the unit ceases further processing. If the command is CUE, the finished state is achieved when the unit fully cues the clip and is ready to playout. If the command is PLAY, the complete state is achieved reached when playback finishes.

If no error occurs in the unit before the desired sync state is reached, the response code is 200 (or 201, or 202, as appropriate). If an error occurs before the unit reaches the desired state, an error response code returns.

The default command synchronization mode for a unit may be changed using the USYN command. Or, the default may be overridden on a individual command basis by prepending a forward slash (/) and the sync mode to the unit command. For example, the command "PLAY U5" uses the default sync mode, while the command "/SYNC PLAY U5" specifies that the command response should not be sent until the PLAY command completes execution.

COMMAND SEQUENCING

Command sequencing specifies how consecutive commands are processed by a unit following successful parse and queue by the unit command buffer. There are two variables which determine how a command is issued and processed: preemption mode and queuing mode.

PREEMPTION

Preemption mode determines if a subsequent command preempts the previously issued one or if the unit suspends command buffer processing until the previous command in process reaches the execution state. Preemption is controlled by both the "previous" command and the "next" command.

The "previous" command may be issued with a qualifier that defers execution of the "next" command until the "previous" command reaches a specified status. The "next" command may be issued with a qualifier that defers its execution until the "previous" command has reached a certain status.

The preemption modes available for the "previous" command are **NNO** (no deferral), **NINI** (defer next until Init), **NRUN** (defer next until Running), **NCMP** (defer next until Complete).

The preemption modes available for the "next" command are **PNO** (no deferral), **PINI** (defer until previous Init), **PRUN** (defer until previous Running), **PCMP** (defer until previous Complete). **POVR** may also be used to override the defer-next preemption mode of the "previous" command.

NOTE: The **IMM** and **SEQ** qualifiers have been superceded by the new preemption qualifiers, but are still supported for compatibility. **IMM** corresponds to **PNO** and **SEQ** corresponds to **PCMP**.

QUEUING

The queuing mode determines whether a command is placed at the beginning of the unit command queue (making it the next command to be executed), placed at the end of the command queue (making it the last command), or whether the command queue is flushed before adding the new command. The

available queuing modes are prepend (**PRE**), append (**APP**), and flush (**FLSH**).

The default command sequencing is flush/no-defer-next/no-defer-previous, meaning that when an MVCP command is issued, the unit command queue is flushed and the command preempts whatever command is currently being executed by the unit.

The default command sequencing mode for a unit may be changed using the USYN command. Or, the default may be overridden on a individual command basis by prepending a forward slash (/) and the sequencing mode to the unit command. For example, the command "PLAY U5" uses the default sequencing mode, while the command "/APP PLAY U5" specifies that the command should be appended to the unit's command queue.

The default preemption and queuing modes are overridden individually. For example, "/APP /PCMP PLAY U5" places the PLAY command at the end of the command queue and the command does not begin executing until the previous command has completed.

Three common sequencing combinations can be abbreviated:

/SEQA is equivalent to /APP /PCMP.

/IMMP is equivalent to /PRE /PNO.

/IMMF is equivalent to /FLSH /PNO.

COMMAND TRIGGERS

More precise control of command execution can be accomplished by specifying the time at which a given command should begin execution. This feature is supported only for certain commands such as PLAY, REC, and STOP.

TIME-OF_DAY TRIGGERING

To specify a command time, at the beginning of the command line, include an at-sign (@) followed by the desired time-of-day. Several formats for specifying time are understood:

The timecode form, HH:MM:SS:FF may be used to specify the time relative to midnight local time calculated using current frame rate (which is initially set according to the system control vtr.main.timing_standard and can be set with FRAT). For instance, 14:14:00:00 specifies that the command should start at exactly 2:14pm local time.

The ISO 8601 date/time form, yyyymmddThhmmss.ssssssZ, may be used to specify the command time.

Finally, the time-of-day form, SEC.USEC, may be used to specify the time using a more traditional UNIX timebase. SEC is the number of seconds since the standard Epoch, Jan 1, 1970. USEC is microseconds.

Example:

```
LOAD U1 a/COMM OUT
/SEQA CUE U1
```

```
    /SEQA @17:31:00:02 PLAY U1
```

NOTE: If the system is receiving house timecode (via Little Red), command timing will be based on the incoming time; otherwise, the system time will be used.

RESPONSE CODES

The MVCP processor responds to all commands with a single response header line. Certain commands additionally return a single response data line. Other commands return multiple response data lines terminated by a single blank line.

The format of the response header line is a three-digit response code followed by an informational string. Some examples:

```
    200 OK
    410 Invalid clip time
    500 Server error
```

The response codes are divided into the following categories.

```
    2XX Successful command execution
    4XX Command format or setup error
    5XX Command execution error
```

The 2XX success codes include:

```
    200 OK      (Success with no additional data)
    201 OK      (Success with N data lines followed by a blank line)
    202 OK      (Success with one data line)
```

Numeric codes for error return values may be overloaded with distinct error text messages. The 4XX format error codes include:

```
    401 Port name missing
    402 Unit name missing
    403 Port not found
    404 Access mode missing or invalid
    405 Clip name missing
    406 Command not supported
    407 Load mode missing or invalid
    408 <not used>
    409 <not used>
    410 <not used>
    411 Event function missing or invalid
    412 Clock time missing or invalid
    413 Numeric argument missing or invalid
    414 Invalid direction
    415 <not used>
    416 Insertion id missing or invalid
    417 Event insertion failed
    418 Invalid speed
    419 Invalid number of passes
    420 Must specify start time
    421 Invalid load flag(s)
    422 Invalid command sync
```

423 Invalid protection
424 Missing or invalid sort order
425 Missing or invalid time
426 Clip monitor not active
427 Invalid parameter format
428 Node type missing or invalid
429 Invalid event type
432 Too many units specified
433 Invalid timestamp format
434 Frame-rate missing
435 Filename missing
436 Still-frame mode invalid
437 Cannot specify both in and out as relative
439 Sync modifier not supported for this command
440 Invalid frame-rate
441 Invalid target command id
442 Invalid count
443 Invalid unload flag
444 Channel name missing
445 Channel opened on different port (<portname>)
446 Channel not open
447 Channel name missing
448 Too many arguments
449 Clip not open
450 Clip name missing
451 Clip already open
452 Format name missing
453 Source clip name missing
454 Invalid track mask
455 Invalid clip source update mode

456 Invalid clip destination update mode
457 User name missing
458 Password missing
459 Already logged in as <username>
460 Must provide USERname first
461 Must login with USERname and PASSword
462 Subsystem name missing
463 Invalid time channel

The 5XX execution error codes include:

500 General error (message varies)
510 General error (message varies)
511 General error (message varies)
530 Login incorrect

rotatelogs(1M)

NAME

rotatelogs - Rotate MSB log files

SYNOPSIS

/usr/vtr/bin/rotatelogs [**-b** *base-log-name*] [**-c**] [**-d** *log-directory*] [**-l** *daemon*] [**-m** *max-logs*] [**-s** *log-size*] [**-v**]

DESCRIPTION

The **rotatelogs** command is used to end the current MSB log file, by default named */usr/vtr/adm/logs/vtrlog,* and start a new log. The completed log file is moved to a file named */usr/vtr/adm/logs/vtrlog.backup.1*, or the next number in sequence. A series of completed log files will be maintained, by default up to 10, with the oldest being deleted as a new log file is completed.

Typically **rotatelogs** is run out of **cron(1M)** each night and is not invoked directly by the user.

rotatelogs has the following options:

-b *base-log-name*
: Set the base name of the log file to *base-log-name*. The default is *vtrlog*.

-c
: Compress the completed log file. **gzip(1)** is used for the compression. The resulting completed log files will have the form */usr/vtr/adm/logs/vtrlog.backup.*.gz* and can be decompressed with **gunzip(1)**.

-d *log-directory*
: Set the directory of the log files to *log-directory*. The default is */usr/vtr/adm/logs*.

-l *deamon*
: Sets which deamon process to notify. The only current value is the default setting of *vtrd*.

-m *max-logs*
: Maintain up to *max-logs* completed log files. If more than *max-logs* completed log files exist, the oldest will be deleted. The default is 10.

-s *size*
: Do not complete the current log file unless it has reached a size of *size*. The default is 10485760 bytes which is equal to 10 MB.

-v
: Run with verbose debugging output. Normally **rotatelogs** produces no output on sucessful operation.

SEE ALSO

vst(1), *vvtr*(1), *vtrd*(1)

vst(1)

NAME

VST - Video Server Toolkit

SYNOPSIS

Video Server Toolkit is a software platform enabling developers to construct high-performance, scalable video server applications on Silicon Graphics servers.

DESCRIPTION

The core VST software provides management of a simple database of clips (the **Clip Cache**), a control API for managing and operatingVST, and a core library which supports various **External Interface Modules**.

A single VST instance manages all playback and record operations for a single video server. However, each VST instance supports multiple logical playback/record units. Each unit manages a single signal port but may be controlled by multiple control ports. A signal port may support multiple logical units subject to the device-sharing characteristics of that port.

External Interface Modules

The VST interface modules fall into five categories:

Storage
: The storage interfaces provide access to the storage systems where the clip content is stored (the **Clip Cache**). The XFS-based disk interface module provided with the base VST software is sufficient for most applications.

Media
: The media I/O interfaces provide access to the "signal" ports of the video server. A signal port is any video or data networking I/O port over which the digital media will be transmitted or received. Each type of I/O port will typically have its own VST interface module.

Format
: The format interfaces provide handlers for accessing specific digital media storage formats. For instance, the base "stream" format handler is suitable for stream-based formats such as MPEG. The "vframe" format handler is specialized toward variable-length, frame-oriented video/audio material.

Control
: The control interfaces implement external control protocols to allow integration with a variety of automation controllers and digital media applications. The VST core control logic provides a wide variety of control capabilities which should be sufficient to implement most machine control protocols.

Archive
: The archive interfaces provide support for content archiving (both

storage and retrieval) on external archive systems.

Control Interfaces

The base VST software three external control protocols:

Louth Video Disk Communications Protocol
: This protocol, defined by Louth Automation, provides full-featured control of the VST via RS-232, RS-422, and TCP/IP. The VST Louth processor supports back-to-back play and record (subject to restrictions imposed by the video I/O port capabilities) and archive management.

Multiport Video Computer Protocol (MVCP)
: This protocol, defined by Silicon Graphics, provides full-featured control of VST via TCP/IP. The protocol processor supports archive management, multiple-unit control, and event monitoring. The protocol command set is documented in mvcp(5).

Industry-standard RS-422 VTR protocol (a.k.a. Sony protocol)
: This protocol, implemented in varying flavors by nearly all broadcast-quality VTRs, provides VTR-emulation capability via RS-232 and RS-422.

SEE ALSO

vtrstart(1), *vtrstop*(1), *vtrstat*(1), *vtrclip*(1), *mcpanel*(1), *mcclips*(1), *mcstat*(1), *mccompstats*(1), *vst-controls*(5), *mvcp*(5), *vvtr*(1), *vtrd*(1), *vtrvfutil*(1)

vst-controls(5)

NAME

vst-controls - Media Server For Broadcast (MSB) Controls

DESCRIPTION

MSB controls parameterize select system values and codec registers, governing operational behavior. These parameters may be set at startup through an initialization file specific to a given MSB subsystem. The control parameters may also be interactively modified via the MVCP SSET command, or queried via the SGET command following MSB initialization startup.

SYSTEM CONTROLS

MSB controls are organized into specific subsystems. These subsystems are identified by the classes: clipmirror, main, filesystem (fs), time, and devices. The devices subsystem is further sub-classed into filesystem controls and codec controls.

The clipmirror class furnishes controls to activate and manipulate clip cache mirror operation and execution. The clipmirror subsystem furnishes a redundant storage mechanism for clip content.

The filesystem class furnishes controls to activate and manipulate the persistent clip storage mechanism, an integral element of MSB infrastructure. The filesystem enables real-time playout, record, edit, and ftp transfer of clip content.

The device class furnishes controls to configure and manipulate video codec operation. The video codec processes video and audio signals, generating clip content during playout, or capturing the real-time feed during record and transforming into digital persistent storage.

NOTE: There are many controls to govern MSB execution behavior. Appropriate SGET queries will reveal their names and current values; SSET can be used without constraint to modify them. **However, unexpected and potentially catastrophic MSB behavior may arise if controls are set which are not explicitly documented in this page.**
Further, the control values assigned by either the initialization file, or via the MVCP SSET are not subject to rigorous range checking or verification. MSB will not honor an invalid input control value, and *may* arbitrarily and silently assign a suitable default. Control value range checking is not rigorously performed in general.

A system-defaults file resides in /usr/vtr/config/system-defaults and is named according to the name of the subsystem.

The following subsystems and controls are currently supported:

SUBSYSTEM: clipmirror
Defaults file: /usr/vtr/config/system-defaults/clipmirror

vtr.clipmirror.primary_server.hostname
Default: "" (null string; mirroring disabled)

Specifies the primary MSB server clip cache to mirror. Setting this value activates clip mirror processing. Setting this value to "" disables clip mirror processing. This control value must be set on the clip mirror platform.

vtr.clipmirror.max_threads
Default: 20

Specifies the number of concurrent ftp transfers supported by the clip mirror platform. This value should consider available network bandwidth between the primary MSB server and mirror platforms. The permissible range for this control is 1 to 100. This control value must be set on the clip mirror platform.

vtr.clipmirror.reconnect_interval
Default: 30

Specifies how often (in seconds) the clip mirror platform attempts to reconnect with the primary MSB server if the connection terminates. This control value must be set on the clip mirror platform.

vtr.clipmirror.local_server.hostname
Default: "" (null string)

Contains the named identity of the primary MSB network interface dedicated for file transfer. Set on the primary MSB platform. Setting this value causes clip mirror transfers from the primary MSB server to the clip mirror to occur via the named interface. For example, if the primary MSB server possesses two network interfaces, one of them might be a 100BaseT ethernet (and known by the name vst1-enet) and the other might be a FibreChannel connection (and known by the name vst1-fc). If the default hostname is vst1-enet but vst1-fc is a higher bandwidth connection, then clip transfer can be assigned to vst1-fc.

SUBSYSTEM: Filesystem
Defaults file: /usr/vtr/config/system-defaults/fs

vtr.storage.fs.grio.bandwidth.copy.enable
Default: false

Enables or disables use of Guaranteed Rate I/O (GRIO) rate guarantees for file system copy operations. If this control is set to true then any file that is copied to or from the real-time file *system*(s) under the mount point /usr/vtr/clips will use a GRIO reservation. If this control is set to true then the controls:

vtr.storage.fs.grio.bandwidth.copy.perstream
vtr.storage.fs.grio.bandwidth.copy.total

need to be set in accordance with the instructions for these controls. This control enables bandwidth management for MVCP commands CCP, CBLD as well as for other control protocols.

vtr.storage.fs.grio.bandwidth.copy.total
Default: 0

Sets the total bandwidth that can be used for copy operations for each file system mounted under /usr/vtr/clips. Once set then any copy operation that exceeds the total copy bandwidth will fail. This control can be used to limit the total number of copy operations so that system resources remain available for video play and record functions. This setting should be set at a fraction of the total bandwidth available to the individual file systems. This setting does not control bandwidth allocated to ftp sessions.

vtr.storage.fs.grio.bandwidth.copy.perstream
Default: 0

Sets the bandwidth that can be used for copy operations for each file. This setting is set to a fractional value of the total bandwidth available for copy operations. If a copy is made within the clip cache then two reservations will be made. One reservation for the read operation and one reservation for the write operation. If any reservation exceeds the total bandwidth available for copy operations then the copy will fail.

vtr.storage.fs.latency_warning_threshold
Default: 400

Specifies the threshold (in milliseconds) which if exceeded by a single I/O operation during playback or recording will log a warning message.

vtr.storage.fs.max_cue_reservations
Default: 1

Specifies the maximum number of bandwidth reservations allowed per real-time file system for cueing clips for playback. A cueing reservation allocates one I/O operation per second. If more units are cueing than the limit specified by this control, the units share the reservation round-robin.

SUBSYSTEM: main
Defaults file: /usr/vtr/config/system-defaults/main

vtr.main.log_level.console
Default: 0

Specifies the maximum log level of messages sent to stdout.

vtr.main.log_level.file
Default: 0

Specifies the maximum log level of messages sent to the server log file (usually, /usr/vtr/adm/logs/vtrlog).

vtr.main.log_level.syslog
Default: 0

Specifies the maximum log level of messages sent to the system log (usually /var/adm/SYSLOG).

vtr.main.timing_standard
Default: 525

Specifies the default timing standard for the system. Certain subsystems and devices use this value for determining the system timing standard. For example, MVCP initializes the control frame rate (FRAT command) using the value of this control. The initial output timing of the MFCODEC device is set based upon this control if the device timing control is set to "system".

This default is also used in certain situations when there is ambiguity about the timing standard in use. For example, certain DVCPRO DIF files may not have the timing standard information available in the headers. This control may be set to either "525" or "625" to select the default behavior.

SUBSYSTEM: time
Defaults file: /usr/vtr/config/system-defaults/time

vtr.time.slave_system_time
Default: true (for channel 1 unless another channel specifically enabled)

Specifies whether the system time (maintained by IRIX) is slaved to the timecode inputs for this time channel. Only one channel can be used to slave the system time.

Depending on the stability of the incoming timecode signal, the IRIX system time usually tracks within 500us of the input timecode.

vtr.time.offset
Default: -27000000 (Miranda Little-Red), -6600000 (Horita PR-232)

Specifies the offset (in nanoseconds) between the actual timebase and the decoded input timecode. This offset accounts for delays in the timecode reader, in the serial communications with VST, and in the VST time processing.

DEVICE CONTROLS

Device controls are configuration, control, and status variables which are associated with a particular unit instance of a particular device interface. Device controls can be set at startup through the use of a device-defaults file. In addition, a unit's device controls can be set and queried through the MVCP SET and GET commands.

A device-defaults file resides in /usr/vtr/config/device-defaults. Device-defaults files are loaded according to a hierarchical scheme enabling common control settings to be shared among devices.

When a unit device interface is initialized, control settings are loaded from device-defaults files in the following order:

Settings for all devices:	ALL
Settings for node type:	MEDIA
Settings for port type:	VIDEO, DECK
Settings for device class:	<device-class>

```
 Settings for device:           <device-name>
```

For example, when the unit device interface for the MFCODEC_1 video device is initialized, settings are loaded from the device-defaults files as follows:

```
 ALL
 MEDIA
 VIDEO
 MFCODEC
 MFCODEC_1
```

The following device controls are currently supported:

DEVICE: Diaquest (deck-control)
Defaults file: /usr/vtr/config/device-defaults/dq[_n]

vtr.deck.error
    Default: (none)

    This read-only control returns the last error detected from the deck control interface.

vtr.deck.status
    Default: (none)

    This read-only control returns the current deck transport status.

vtr.edit.coincidence.preroll
    Default: :10

    Specifies how far in advance of the edit in-point that VST attempts to sync. Under some circumstances, the actual coincidence point will be a few frames later than specified, so this control should be set to at least 10 frames.

vtr.edit.preroll
    Default: 5:00

    Specifies the deck preroll for edit operations.

vtr.edit.postroll
    Default: 3:00

    Specifies the deck postroll for edit operations.

vtr.edit.status
    Default: (none)

    This read-only control returns the current status of an edit operation that is in-progress.

vtr.media.video.frame_rate
    Default: (none)

    This read-only control returns the video frame rate currently being reported by the deck.

vtr.media.video.sync_port
Default: (none)

Specifies the port number from which the deck control interface should derive video sync timing. This should be set to the port that will be used for capturing from or laying-down to the deck.

If the control is not set, VST will use any available port which has active video, but a warning will be issued.

vtr.media.video.input.sync.offset.fields
Default: 0

Specifies the number of fields to be used as the video sync offset when capturing from the deck. If captures from the deck start and end too late, a negative offset can be used to move the synchronization earlier. If the captures start and end too early, a positive offset should be used.

vtr.media.video.output.sync.offset.fields
Default: 0

Specifies the number of fields to be used as the video sync offset when laying-down to the deck. If lay-downs to the deck start and end too late, a negative offset can be used to move the synchronization earlier. If the lay-downs start and end too early, a positive offset should be used.

vtr.media.output.mode
Default: pb

Specifies the current output mode of the deck. The value may be "pb" for normal playback or "ee" for E-to-E.

DEVICE: Filesystem
Defaults file: /usr/vtr/config/device-defaults/fs

vtr.storage.clear_after_play
Default: false

Specifies that frames are automatically cleared from the clip after being played. The clip must be loaded for input/output.

This mechanism is used to implement a delay. One unit records the clip while another unit plays the same clip delayed by some fixed interval. Setting the clear_after_play control releases the disk space occupied by the frames after they have been played; hence, the disk utilization remains constant.

This control takes effect immediately.

vtr.storage.continue_after_error
Default: true

Specifies that the unit continues playing after an I/O error occurs. Any frames that could not be read from the clip file will be played as black.

vtr.storage.io_log_level
Default: 4

Specifies the log level for I/O logging messages.

vtr.storage.max_open_files
Default: 8

Specifies the maximum number of clip files that the unit may have open at a single time. If the unit is playing a segmented clip with a large number of short source clips, the source clips will be opened and closed as needed to stay within the specified limit.

vtr.storage.read.contiguous_threshold
Default: 16384 (bytes)

Specifies the maximum allowable interframe gap (the distance between the ending of one frame and the beginning of another). If the interframe gap is less than the value of this control, vst will perform an extent size io read instead of a single frame io read.

DEVICE: MFCODEC
Defaults file: /usr/vtr/config/device-defaults/MFCODEC[_n]

vtr.media.audio.input.channel_map.source
Default: EEEA

Specifies which audio input source is used for each audio channel pairs in recording. 'E' for embedded, 'D' for AES/EBU, 'A' for analog input.

If recording is in-progress when this control is changed, the input audio device will not be reconfigured until recording is stopped and restarted.

vtr.media.audio.input.channel_map.channel_pair
Default: 1234

Specifies audio input channel pairs in the order to be recorded in the clip. For example, a setting of "1111" will record the first pair to all the audio channels in the clip.

If recording is in-progress when this control is changed, the input audio device will not be reconfigured until recording is stopped and restarted.

vtr.media.audio.input.channels
Default: 4

Specifies the number of audio input channels to be read from the audio device. This can be fewer than the number of channels supported by the audio device but cannot be greater. Currently, the supported values are 1, 2, and 4.

If recording is in-progress when this control is changed, the input audio device will not be reconfigured until recording is stopped and restarted.

vtr.media.audio.input.sync.offset
Default: 0

Specifies the offset (in milliseconds) to be applied to the audio being ingested. This value is typically determined after measuring a-v delay, and can be positive or negative.

vtr.media.audio.output.analog_channel
Default: 1

Specifies the audio channel pair to be played out on the analog output. A value of 13 causes LTC output to be inserted on analog output.

If playback is in-progress when this control is changed, the output audio device will not be reconfigured until recording is stopped and restarted.

vtr.media.audio.output.max_embedded_channels
Default: 8

Specifies the maximum number of audio channels to embed in the outgoing SDI video signal (value can be either 4 or 8).

vtr.media.audio.output.start_offset
Default: 0

Specifies the number of video fields to offset the start of audio playback. This value should be 0 for MPEG and -1 for DVCPRO.

vtr.media.video.input.constant_bit_rate
Default: true

Enables constant bit rate encoding. If set to false, variable bit rate will be used.

If recording is in-progress when this control is changed, the setting will not be changed until recording is stopped and restarted.

vtr.media.video.input.format
Default: sdi

No other input format is currently supported.

vtr.media.video.input.field_dominance
Default: F1

Specifies the input field dominance. Possible values are F1 and F2.

This control is only read when a new clip is being created. Changing this control will not affect any existing clips including the one currently loaded.

vtr.media.video.input.hard_reset
Default: false

Setting this control will reset the video device. All recording in

progress on this device will be stopped. This control should only be used as a last resort in error recovery.

vtr.media.video.input.horizontal_res
Default: 720

Specifics the horizontal resolution of the video signal to be captured. '720' is the only supported value in this release.

vtr.media.video.input.vertical_res.extended
Default: true

Enables encoding of the entire video signal, including the vertical blanking interval. When set to "true", in NTSC, 512 lines of video signal is captured; and in PAL, 608 lines of video signal is captured. When set to "false", in NTSC, only 408 lines of video signal is captured; and in PAL, 576 lines of video signal is captured.

If recording is in-progress when this control is changed, the setting will not be changed until recording is stopped and restarted.

vtr.media.video.input.vitc.line_offset
Default: 14

Specifies the video line where vitc (vertical interval time code) data is present in the video signal. The default is line 14 and 16.

If recording is in-progress when this control is changed, the setting will not be changed until recording is stopped and restarted.

vtr.media.video.output.vitc.line_offset
Default: 14

Specifies the video line where vitc (vertical interval time code) data is inserted in the video signal. The default is line 14 and 16.

If playback is in-progress when this control is changed, the setting will not be changed until playback is stopped and restarted.

vtr.media.video.output.hard_reset
Default: false

Setting this control will reset the video device. All play out in progress on this device will be stopped. This control should only be used as a last resort in error recovery.

vtr.media.video.output.image.name
Default: black

Specifies the name of the image displayed when the unit output mode is "image" and the vtr.media.output.image.type is "user". The image must be placed in the appropriate directories under /usr/vtr/data/images.

vtr.media.video.output.image.type
Default: bars

Specifies the image displayed when the unit output mode is "image". The possible values are:

smpte-75-bars	75% saturation SMPTE color bars with PLUGE
75-bars-over-red	75% saturation color bars over red
bars	smpte-75-bars (525), 75-bars-over-red (625)
black	Black
user	User (vtr.media.video.output.image.name)

vtr.media.video.output.jog.fields

Default: false

Setting this control effects how media MSB jogs (for example single steps) through video. The choices are:

false	Jog in frames, displaying the top field.
true	Jog in fields.

vtr.media.output.cued_mode

Default: clip

Specifies the unit's output mode when it is cued. The possible values are:

clip	Normal clip video/audio output
hold	Keep frame from last played clip
black	Black (silent audio)
image	Still image (vtr.media.video.output.image.type)

vtr.media.output.goto_mode

Default: clipOnly

Specifies the unit's behavior when a "GOTO timecode" is handled. The possible values are:

clipOnly	Accept timecode within the clip; ignore timecode outside of the clip start/end boundaries and issue an error message in the logfile
inoutOnly	Only accept timecode within hard/soft IN/OUT points of the clip if these exist; ignore the timecode if it is outside of the hard/soft IN/OUT point and issue an error message in the logfile
inoutClamp	Accept timecode within the clip; if the timecode is within the clip start/end boundaries but outside of hard/soft IN/OUT points, clamp it to IN/OUT points; issue an error message in the logfile if the timecode is outside of the clip start/end boundaries

vtr.media.input.trigger.mode.in

Default: none

Specifies the type of trigger used to begin recording. The default value, "none", disables triggering; recording begins immediately (subject to command scheduling if the record command is time-stamped).

The value "vitc" enables VITC-triggering. When the unit record command is received, the unit begins monitoring the VITC of each

input frame. When an input frame whose VITC matches the value of **vtr.media.input.trigger.vitc.in** is received, recording begins. The triggering frame is the first frame recorded to the clip.

The value "ltc" enables LTC-triggering. When the unit record command is received, the unit begins monitoring the LTC of each input frame. When an input frame whose LTC matches the value of **vtr.media.input.trigger.ltc.in** is received, recording begins. The triggering frame is the first frame recorded to the clip. LTC input is expected to be on the analog audio input jack.

This control is only read once before the recording is started.

vtr.media.input.trigger.mode.out
Default: none

Specifies the type of trigger used to end recording. The default value, "none", disables triggering; recording ends immediately after the unit receives a stop command (subject to command scheduling if the record command is time-stamped).

The value "vitc" enables VITC-triggering. When recording begins, the unit begins monitoring the VITC of each input frame as it is recorded. When an input frame whose VITC matches the value of **vtr.media.input.trigger.vitc.out** is received, recording ends.

The value "ltc" enables LTC-triggering. When recording begins, the unit begins monitoring the VITC of each input frame as it is recorded. When an input frame whose LTC matches the value of **vtr.media.input.trigger.ltc.out** is received, recording ends. LTC input is expected to be on the analog audio input jack.

This control is only read once before the recording is started.

vtr.media.input.trigger.vitc.in
Default: (none)

Specifies the VITC of the first frame recorded when the input trigger-in mode is VITC.

This control is only read once before the recording is started.

vtr.media.input.trigger.vitc.out
Default: (none)
Specifies the VITC of the frame following the last frame recorded when the input trigger-out mode is VITC.

vtr.media.input.trigger.ltc.in
Default: (none)

Specifies the LTC of the first frame recorded when the input trigger-in mode is LTC.

This control is only read once before the recording is started.

vtr.media.input.trigger.ltc.out
Default: (none)

Specifies the LTC of the frame following the last frame recorded when the input trigger-out mode is LTC.

This control is only read once before the recording is started.

vtr.media.input.timecode.mode
Default: source

Specifies the timecode to be stored in each DIF frame during DVCPRO capture. When a DVCPRO clip is played back, this time code is outputted when the vitc or ltc output mode is set to "source" (see vtr.media.video.output.vitc.mode). The possible values are:

off	Zeroed
source	VITC timecode
clip	Clip location (CTL) timecode
vitc	VITC timecode
ltc	LTC timecode
freerun	Value of free-running counter
run	Value of running counter
date	Current date
time	Current time
constant	Constant value

The free-running counter begins incrementing immediately from the value set into **vtr.media.input.timecode.preset** and continues running regardless of whether the unit is recording or idle.

The running counter increments from the value set into **vtr.media.input.timecode.preset**, but only when the unit is actively recording.

The constant value is also determined by the timecode preset value.

This control takes effect immediately.

vtr.media.input.timecode.preset
Default: 0

Specifies the initial value of the free-running counter which is used by **vtr.media.input.timecode.mode** in freerun, run and constant modes. The value must be a valid timecode.

This control takes effect immediately.

vtr.media.input.userbits.mode
Default: source

Specifies the userbits to be stored in each DIF frame during DVCPRO capture. When a DVCPRO clip is played back, these userbits are outputted when the userbit output mode is set to "source" (see vtr.media.video.output.vitc.userbits.mode). The possible values are:

off	Zeroed
source	VITC userbits
clip	Clip location (CTL) timecode

vitc	VITC userbits
ltc	LTC userbits
freerun	Value of free-running counter
run	Value of running counter
date	Current date
time	Current time
constant	vtr.media.input.userbits.constant

The free-running counter begins incrementing immediately from the value set into **vtr.media.input.userbits.preset** and continues running regardless of whether the unit is recording or idle.

The running counter increments from the value set into **vtr.media.input.userbits.preset**, but only when the unit is actively recording.

For both the running and free-running counter, the preset value used must be a valid time code. To insert an arbitrary constant into the userbits use the choice "constant" and load the value with **vtr.media.input.userbits.constant**.

This control takes effect immediately.

vtr.media.input.userbits.preset
Default: 0

Specifies the initial value of the free-running counter which is used by **vtr.media.input.userbits.mode** in freerun and run modes. The value must be a valid timecode.

This control takes effect immediately.

vtr.media.input.userbits.constant
Default: 0

Specifies an arbitrary constant for insertion into the userbits of each DIF frame during DVCPRO capture. This constant should be in hex format, for example, 0xaabbccdd. This constant is selected by the setting the control vtr.media.input.userbits.mode to "constant".

This control takes effect immediately.

vtr.media.clip.format
Default: default

Specifies the format of new clips that are created. The values currently supported are:

default	native DV and MPEG2
movie/stream/mxf	MXF (DVCPro and MPEG2)

If "default" is specified, the appropriate format for the compression type specified by vtr.media.video.input.compression.type is selected.

This control is only read when a new clip is being created. Changing this control will not affect any existing clips including the one currently loaded.

vtr.media.clip.start.preset
Default: 01:00:00:00

Specifies the timecode of the first frame recorded into an empty clip if no timecode is specified when the unit is cued for recording and the vtr.media.clip.start.mode is set to "preset".

This control is only read when a newly created clip is cued for recording and no in-point has been specified for cueing.

vtr.media.clip.start.mode
Default: preset

Specifies the mode for determining the timecode of the first frame recorded into an empty clip if no timecode is specified when the unit is cued for recording. A value of "preset" specifies that the timecode is taken from vtr.media.clip.start.preset. A value of "time-of-day" specifies that the timecode is taken from the current time-of-day when the first frame comes into the system. A value of "vitc" specifies that the timecode is taken from the VITC embedded in the first frame. A value of "ltc" specifies that the timecode is taken from the LTC corresponding to the first frame.

This control is only read when a newly created clip is cued for recording and no in-point has been specified for cueing.

vtr.media.clip.start.preroll.ltc
Default: 4

Specifies the preroll used when vtr.media.clip.start.mode is set to "ltc". This control pertains only to mpeg capture and should be changed with great care.

vtr.media.clip.start.preroll.vitc
Default: 3

Specifies the preroll used when vtr.media.clip.start.mode is set to "vitc". This control pertains only to mpeg capture and should be changed with great care.

vtr.media.mpeg.bit_rate
Default: 50000000

Specifies the bit rate used in encoding video. If vtr.media.video.input.constant_bit_rate is set to false, this will be the maximum bit rate used.

If recording is in-progress when this control is changed, the setting will not be changed until recording is stopped and restarted.

vtr.media.video.port.mode
Default: output

Specifies whether the device should be configure as an output port or an input port. Valid settings are "output" and "input".

If a play or record is in-progress when this control is changed, the

setting will not be changed. Changing this setting will cause a reload of system firmware. /SYNC should be used to ensure that the system has finished reloading the new firmware.

vtr.media.video.input.compression.type
Default: mpeg2

Specifies the type of compression applied to video frames being recorded.

The possible values are:

mpeg2	MPEG2 encoding
dvcpro	DVCPRO-25 encoding
dvcpro50	DVCPRO-50 encoding

If recording is in-progress when this control is changed, the setting will not be changed. Changing this setting will cause a reload of system firmware. /SYNC should be used to ensure that the system has finished reloading the new firmware.

vtr.media.video.input.compression.sampling
Default: 422

Specifies the sampling rate used in encoding the video signal. Valid values are "422" and "420". This control only has an effect when compression type is set to "mpeg2".

If recording is in-progress when this control is changed, the setting will not be changed. Changing this setting will cause a reload of system firmware. /SYNC should be used to ensure that the system has finished reloading the new firmware.

Note: 420 compression sampling is limited to 15Mbit/sec bit rate by the PCI-VIDAUD-MSB hardware. The **vtr.media.mpeg.bit_rate** values exceeding this limit are ignored under 420 compression sampling. 420 compression may generate large block-pixel images, depending on the ingested content scene change rate.

vtr.media.video.input.compression.gop_structure
Default: I

Specifies the gop structure for encoding when the compression type is set to "mpeg2". The gop structure determines the type and pattern of MPEG frames (I, P, or B) that are used within a GOP.

The possible values are:

I	GOP period = 1
IP	GOP period = 1
IPB	GOP period = 2
IPBB	GOP period = 3

The GOP period of a structure must evenly divide the GOP size (see below). "IPBB" encoding is only available when the compression sampling is set to "420".

vtr.media.video.input.compression.gop_size
Default: 1

Specifies the size of the GOP that is used when encoding video. The gop size must be a multiple of the the gop period (as implied by the control **vtr.media.video.input.compression.gop_structure**. The GOP size must be between 1 and 15 (inclusive). For a given gop size and period some sample GOPS would be:

size, period	Stored GOP
1,1	I
15,1	IPPPPPPPPPPPPPP
6,3	IBBPBB
15,3	IBBPBBPBBPBBPBB

vtr.media.video.input.picture_rate
Default: 29.97

Specifies the timing of the video input signal. "29.97" denotes NTSC. "25" denotes PAL.

If recording is in-progress when this control is changed, the setting will not be changed. Changing this setting will cause a reload of system firmware. /SYNC should be used to ensure that the system has finished reloading the new firmware.

vtr.media.audio.output.backpanel
Default: true

Specifies the back panel is installed. If dangle is used for audio input and/or output, vtr.media.audio.output.backpanel should be set to false.

vtr.media.video.output.vitc.mode
Default: source

Specifies the data sent for the VITC time code of each frame. The possible values are:

off	Zeroed
source	VITC from recorded clip
clip	Clip location (CTL) timecode
vitc	VITC from recorded clip
ltc	LTC from recorded clip
freerun	Value of free-running counter
run	Value of running counter
date	Current date
time	Current time
constant	Constant value

The free-running counter begins incrementing immediately from the value set into **vtr.media.video.output.vitc.timecode.preset** and continues running regardless of whether the unit is playing or idle.

The running counter increments from the value set into **vtr.media.video.output.vitc.timecode.preset**, but only when the unit

is actively playing.

The constant value is also determined by the vitc timecode preset value.

This control takes effect immediately.

vtr.media.video.output.vitc.timecode.preset
Default: 0

Specifies the initial value of the free-running counter which is used by the VITC time code insertion in freerun, run and constant modes.

This control takes effect immediately.

vtr.media.video.output.vitc.userbits.mode
Default: source

Specifies the data sent in the VITC user bits of each frame. The possible values are:

off	Zeroed
source	VITC userbits from recorded clip
clip	Clip location (CTL) timecode
vitc	VITC userbits
ltc	LTC userbits
freerun	Value of free-running counter
run	Value of running counter
date	Current date
time	Current (local) time
constant	vtr.media.video.output.vitc.userbits.constant

The free-running counter begins incrementing immediately from the value set into **vtr.media.video.output.vitc.userbits.preset** and continues running regardless of whether the unit is playing or idle.

The running counter increments from the value set into **vtr.media.video.output.vitc.userbits.preset**, but only when the unit is actively playing.

For both the running and free-running counter, the preset value used must be a valid time code. To insert an arbitrary constant into the user bits use the value "constant" and enter the desired constant with **vtr.media.video.output.vitc.userbits.constant**.

This control takes effect immediately.

vtr.media.video.output.vitc.userbits.preset
Default: 0

Specifies the initial value of the free-running counter which is used for VITC user bit insertion in freerun and run modes. Must be a valid timecode.

This control takes effect immediately.

vtr.media.video.output.vitc.userbits.constant
Default: 0

Specifies an arbitrary constant for insertion into the vitc userbits. This constant should be in hex format, for example, 0xaabbccdd. To use this constant, a value of "constant" must be specified for vtr.media.video.output.vitc.userbits.mode.

This control takes effect immediately.

vtr.media.video.output.ltc.mode
Default: source

Specifies the data sent for the LTC time code of each frame. The possible values are:

off	Zeroed
source	LTC from recorded clip
clip	Clip location (CTL) timecode
vitc	VITC from recorded clip
ltc	LTC from recorded clip
freerun	Value of free-running counter
run	Value of running counter
date	Current date
time	Current time
constant	Constant value

The free-running counter begins incrementing immediately from the value set into **vtr.media.video.output.ltc.timecode.preset** and continues running regardless of whether the unit is playing or idle.

The running counter increments from the value set into **vtr.media.video.output.ltc.timecode.preset**, but only when the unit is actively playing.

The constant value is also determined by the ltc timecode preset value.

This control takes effect immediately.

vtr.media.video.output.ltc.timecode.preset
Default: 0

Specifies the initial value of the free-running counter which is used by the LTC time code insertion in freerun, run, and constant modes.

This control takes effect immediately.

vtr.media.video.output.ltc.userbits.mode
Default: source

Specifies the data sent in the LTC user bits of each frame. The possible values are:

off	Zeroed
source	LTC userbits from recorded clip
clip	Clip location (CTL) timecode
vitc	VITC userbits
ltc	LTC userbits

freerun	Value of free-running counter
run	Value of running counter
date	Current date
time	Current (local) time
constant	vtr.media.video.output.ltc.userbits.constant

The free-running counter begins incrementing immediately from the value set into **vtr.media.video.output.ltc.userbits.preset** and continues running regardless of whether the unit is playing or idle.

The running counter increments from the value set into **vtr.media.video.output.ltc.userbits.preset**, but only when the unit is actively playing.

For both the running and free-running counter, the preset value used must be a valid time code. To insert an arbitrary constant into the user bits use the value "constant" and then enter the desired constant with **vtr.media.video.output.ltc.userbits.constant**.

This control takes effect immediately.

vtr.media.video.output.ltc.userbits.preset
Default: 0

Specifies the initial value of the free-running counter which is used for LTC user bit insertion in freerun and run modes.

This control takes effect immediately.

vtr.media.video.output.ltc.userbits.constant
Default: 0

Specifies an arbitrary constant for insertion into the ltc userbits. This constant should be in hex format, for example, 0xaabbccdd. This constant is selected by setting the control **vtr.media.video.output.ltc.userbits.mode** to "constant".

vtr.media.video.output.phase.horizontal
Default: 0

Specifies the horizontal offset (in half-pixels) applied to the output video relative to the incoming sync reference. This control is only meaningful if the output video is properly synchronized to an external reference (genlocked).

For example, to offset the output video 20 pixels from the the incoming reference input, a value of 40 would be required. There is no visible shift in the outputted video because it is inserted in the correct active portions of the offset timing. The shift adjustment can be used for SDI video by applications that want the SDI output video to have a timing similar to a house sync.

Only positive values are allowed. This control takes effect immediately.

vtr.media.video.output.sync.offset.vertical
Default: 0

Specifies the vertical offset (in lines) applied to the output video relative to the incoming sync reference. This control is only meaningful if the output video is properly synchronized to an external reference.

This control takes effect immediately.

vtr.media.video.output.compression.type

Default: mpeg2

Specifies the type of compressed media that is initially played through an output port.

The possible values are:

mpeg2	MPEG2 encoding
dvcpro	DVCPRO-25 encoding
dvcpro50	DVCPRO-50 encoding

MPEG2 uses different video settings then the two DVCPRO formats. This means that the hardware must be initialized differently for MPEG2 then for DVCPRO-25/50. As a result, MPEG2 clips cannot be played back-to-back with DVCPRO-25/50 clips using the same port. However, if an output port is not playing then MSB does support switching to a clip of a different format (i.e. switching between MPEG2 and DVCPRO-25/50). This control allows the user to specify the default behaviour of MSB - whether it is initialized to start playing MPEG2 or DVCPRO-25/50 clips.

vtrclip(1)

NAME

vtrclip - MSB clip insertion/deletion utility

SYNOPSIS

/usr/vtr/bin/vtrclip *options* **add** [**-noindex** | **-rebuildindex**] *clip* ...
/usr/vtr/bin/vtrclip *options* **rm** *clip* ...
/usr/vtr/bin/vtrclip *options* **convert** *clip* ...

DESCRIPTION

vtrclip enables clip insert and delete operations. The insert operation permits externally generated media to be introduced to the MSB clip cache. The delete operation permits cache-resident clip content to be safely deleted. The **vtrclip** utility allows users to convert sgi mpeg2 files to gxf files by invoking **vtrmpegutil(1)**.

Clips may be specified either using the clip name or specifying the clip filename path (clip files are stored underneath **/usr/vtr/clips**).

The following general options may be specified:

-quiet
Quietly perform the specified operation. Error messages are not suppressed.

-verbosity *level*
Set the verbosity to the specified *level* (for debugging).

ADDING CLIPS

Clips are added to the clip cache by specifying the **add** argument. The clip media file must reside in the appropriate directory underneath **/usr/vtr/clips**.

vtrclip generates an index (underneath **/usr/vtr/index**) for the clip before adding it to the clip cache unless, a pre-existing index is found, or the **-noindex** option is specified.

If a pre-existing index file is found, **vtrclip** does not rebuild this file unless the **-rebuildindex** option is specified.

REMOVING CLIPS

Clips are removed from the clip cache by specifying the **rm** argument. **vtrclip** unconditionally removes the clip content and any attributed index file.

CONVERTING CLIPS

Sgi MPEG-2 clips are converted to GXF clips by specifying the convert subcommand. **vtrclip** converts the clip without prompting for comfirmation.

NOTES

The MSB clip cache is intimately coupled to MSB execution. This implies that native UNIX filesystem operations against clip cache components, specifically clip content and index files, will corrupt and permanently

perturb the cache structure, possibly leading to adverse or inconsistent MSB operation. It is therefore required that this utility, and others from the MSB command suite, be used exclusively to manipulate clip content elements outside of MVCP sessions. **DO NOT** use rm(1), mv(1), cp(1), or other native UNIX filesystem commands to manipulate clip content residing in the real-time filesystem.

SEE ALSO

vtrmpegutil(1)

vtrd(1)

NAME

vtrd - MSB parent daemon

SYNOPSIS

/usr/vtr/bin/vtrd [**-c** *config-file*] [**-d** *core-dir*]
[**-f**] [**-l** *log-options*] [**-L** *log-file*] [**-s** *log-level*]
[**-v** *log-level*]

DESCRIPTION

The MSB parent daemon is the top-level process of the MSBplatform software. **vtrd** is responsible for starting, stopping, monitoring, and restarting MSB processes in the event of a crash.Under normal circumstances, **vtrd** is not run directly. Use **vtrstart** to start the MSB. Site-dependent options and arguments for **vtrd** should be placed in the file */usr/vtr/config/vtrd.options* which **vtrstart** passes to **vtrd**.

If **vtrd** receives a SIGTERM signal, it will terminate all the processes it is managing, sending each its respective termination signal. Once the processes have all terminated, **vtrd** exits. Under normal circumstances, however, **vtrd** is not directly stopped in this manner. Use **vtrstop** to stop the MSBsystem.

vtrd has the following options:

-D
Debug mode. Run in the foreground. Ordinarily, **vtrd** places itself in the background when it is started.

-c *config-file*
Read the configuration from *config-file*. The format of the configuration file is described below. The default is */usr/vtr/config/vtrd.conf*.

-d *core-dir*
Place process core files under the directory *core-dir*. When a process that **vtrd** has started dumps core, the core file is saved in a directory under *core-dir* with the same name as the process. The default is */usr/vtr/crash*.

-f *log-level*
Set the maximum log message priority for the log file to **Info**+*log-level*. The default is 0, meaning all log messages up to and including **Info** priority are written to the log file.

-F *log-file*
Write the log file to *log-file*. Without this option, no log file is written, even if the **-f** option is specified.

-s *log-level*
Set the maximum log message priority for SYSLOG to **Info**+*log-level*. The default is 0, meaning all log messages up to and including **Info** priority are written to SYSLOG.

-v *log-level*
Set the maximum log message priority for stdout to **Info**+*log-level*. The default is 0, meaning all log messages up to and including **Info** priority are written to stdout.

CONFIGURATION FILE FORMAT

The **vtrd** configuration file, */usr/vtr/config/vtrd.conf*, comprises one or more configuration lines which specify what processes **vtrd** manages and how they are to be started and stopped.

A line that begins with a pound-sign (#) is treated as a comment and ignored. All blank lines are ignored.

A configuration line consists of the following whitespace- delimited fields:

1. process name
2. bound CPU number
3. scheduling priority
4. minimum retry interval
5. maximum retry interval
6. termination signal
7. command

These fields have the following meanings:

process name
Used for logging purposes and for saving core files.

bound CPU number
If specified, the process will be assigned to run only on the specified CPU. Use an asterisk (*) to specify that the process can run on any CPU.

scheduling priority
If specified, the process will be assigned to the real-time FIFO scheduler with the specified priority. Use an asterisk (*) to specify that the process will inherit the same scheduling priority as **vtrd**.

minimum retry interval
Specifies the minimum interval (in milliseconds) between attempts to restart the process if it exits for any reason. If the process exits, **vtrd** will wait this long before restarting it. If it exits again, **vtrd** will extend the wait before restarting the process each time. Use an asterisk (*) to specify the **vtrd** default which is currently 500ms (1/2 sec).

maximum retry interval

Specifies the maximum interval (in milliseconds) between attempts to restart the process if it exits for any reason. If the process repeatedly exits, **vtrd** will wait for longer periods of time between restart attempts. When the wait period reaches the limit specified here, it will not be extended. Use an asterisk (*) to specify the **vtrd** default which is currently 120000ms (2 minutes).

termination signal
Specifies the signal to be used to stop the process when **vtrd** is terminated. A signal number may be specified (e.g., 15) or a signal name (e.g., SIGTERM or TERM).

command
Specifies the executable file and command-line arguments for the process. If the command contains any spaces or other whitespace, enclose the command in double quotation marks.

LOGGING

MSB logs to several destinations. The **-f**, **-s**, and **-v** options are used to adjust the lower bound on the priority of messages logged to each of the log file, system log, and stdout.

The **-F** option can be used to set the name of the log file (the default is no log file). If the **-t** option is used, the log file is truncated each time it is opened; otherwise, the new log messages are appended to the existing log file.

If **vtrd** is sent a SIGHUP signal, it responds by reopening the log file. SIGHUP can be used to rotate log files while **vtrd** remains running. **vtrd** also passes the SIGHUP to each of the processes that it is managing.

SEE ALSO

vcp-recorder(1), *vvtr*(1), *vtrstart*(1), *vtrstop*(1)

vtrdircopy(1)

NAME

vtrdircopy - Directory Copy Daemon for MSB

SYNOPSIS

/usr/vtr/bin/vtrdircopy [*options*]

DESCRIPTION

vtrdircopy is a daemon that monitors a list of source-destination directory pairs. Whenever a file is added to a source directory, **vtrdircopy** will detect the file and ftp it into the corresponding destination directory using **vtrftp**. **vtrdircopy** may be added to **vtrstart** and executed simultaneously with **vvtr**. **vtrdircopy** exits gracefully when signals SIGINT or SIGTERM are posted; The config file **/usr/vtr/config/vtrdircopy.conf** is re-read with a SIGHUP posted.

The config file should contain 3-tuples of the form: *source_path destination_path clip_type*.

source_path must exist on the local server. destination_path may be of the form *remote_server:remote_path*, or may be an existing directory on the local server. *clip_type* should be either I or C, indicating a directory of index files or of clip files.

vtrdircopy has the following options:

-config *config_filename*
: Use the config file *config_filename*, instead of **/usr/vtr/config/vtrdircopy.conf**.

-logfile *log_file*
: Use as log the file *log_file*, instead of **/usr/vtr/adm/logs/vtrdircopylog**. The log file is used by vtrdircopy to recover from crashes and should not be modified by users. It is a text file and may be human-readable, but is not explicitly designed to be conveniently readable.

-procs *num_procs*
: Limit number of spawned child **vtrftp** processes to *num_procs*.

-sync
: The entire source directory is copied over to the destination. Note that this can be a very resource-intensive operation if the source directory is large.

-recover
: This option can be used after a crash to search the log for files that were detected but not ftp'd, or files that started to be ftp'd, but did not finish being ftp'd.

-bandwidth
: Establishes vtrcopy bandwidth constraint. See **vtrcopy** for default.

-indexthresholdsize

Minimum size of index file before copy/ftp commences.

-clipthresholdsize
Mininum size of clip file before copy/ftp commences.

-indexwindowsize
Windowsize for ftp or buffersize for copy.

-failedftp
This option causes **vtrdircopy** to search the log for all failed ftp's and retry those ftps. Before doing this, it is recomended that the cause of the failed ftp's be remedied (failed DNS lookup, inadequate permissions of destination directory, etc). Also, if the log file is large, this may result in a large number of ftp'd files.

-sleep *sleep_between_poll*
vtrdircopy uses a polling mechanism to determine if new files have been added to the source directory. The polling interval may be increased on a heavily loaded system, or decreased to improve response time.

-wait *wait_for_file_growth*
A CUER will create a zero length clip. **vtrdircopy** will wait for *wait_for_file_growth* seconds before trying to transfer a zero length clip. 0 will cause **vtrdircopy** to wait forever until the file becomes non-zero size.

-marker *m*
vtrdircopy uses a marker internally that cannot appear in any pathname (directory or file). The default marker is %.

-ftpdebug *level*
Sets the Perl Net::FTP debug level to *level*.

-verbose
Verbose operation

-help
List of **vtrdircopy** options and what they do

SEE ALSO

vtrftp(1)

vtrfsinfo(1)

NAME

vtrfsinfo - Retrieve MSB filesystem manifest

SYNOPSIS

/usr/vtr/bin/vtrfsinfo [*options*]

DESCRIPTION

vtrfsinfo retrieves MSB file system information for display. This is an internal command, used by VST to acquire basic file system values, such as mount points, iosize, device number, and other fundamental file system attributes. Application of this command for use other than VST internal purposes is not supported.

vtrfsinfo has the following options:

-table *tabular output*
Generates formatted table output; Otherwise, unformatted output is generated.

SEE ALSO

vtrhwinfo(1)

vtrftp(1)

NAME

vtrftp - Command Line Ftp Client for MSB

SYNOPSIS

/usr/vtr/bin/vtrftp [*options*]

DESCRIPTION

vtrftp is a perl ftp client (uses Net::FTP) designed to work with MSB. It does passive transfers between the localhost and the destination host. It does not automatically transfer index files. If you need to transfer a clip and its index file, you should invoke 2 seperate instances of **vtrftp** concurrently. **vtrftp** preserves the mark points (start, in and out) of clips.

Since **vtrftp** only does passive transfers, both the sender and receiver must have vtrftpd daemons available and should be MSB servers with the same version of MSB software. Also, once a transfer is initiated, killing **vtrftp** will not halt the transfer. **vtrftp** only does put. Get is not supported in the current version. File renaming is also not supported.

vtrftp has the following options:

-clip *filename*
Transfer the clip specified by *filename.* This is a mandatory option. The filename may be a relative path to the current working directory, or an absolute path.

-host *hostname*
Transfers clip to the host specified by *hostname.* If this is not specified, localhost is assumed.

-dir *path*
path is the directory on the destination server into which the clip will be put. If this directory does not exist, **vtrftp** will create it. Make sure adequate permissions exist on the destination server for directory creation and file writing. If this option is not used, the file will be put in the ftp root directory on the destination, which is */usr/vtr/clips* for **vtrftpd**. If you do transfers on localhost, you will lose your clip if source and destination directories are the same.

-win *windowsize*
The tcp window size will be set to *windowsize* via the *site win* command. win 0 will use the default window size.

-marks
site marks on **vtrftpd** is invoked before the transfer. For DIF clips, only the content between the mark points will be transferred.

-debug *level*

Sets the Perl Net::FTP debug level to *level*.

-log *level*
vtrftp logs the clips it transfers and their attributes in the file */usr/vtr/adm/logs/vtrftplog*. Set *level* to **1** to enable logging.

-help
List of **vtrftp** options with descriptions

SEE ALSO

vtrftpd(1)

vtrftpd(1)

NAME

vtrftpd(1) - Real-time enabled FTP daemon

SYNOPSIS

/usr/vtr/bin/vtrftpd [ftpd options][-G*bps*][-P*pri*]

DESCRIPTION

vtrftpd is a real-time enabled version of the IRIX FTP daemon, **ftpd**. It provides all the features of the basic **ftpd** plus the following additional capabilities:

+ Stores to the real-time subvolume of a file system.

+ Configurable guaranteed rate I/O for stores or retrieves.

+ Automatic disk- and network-I/O sizing.

+ Honors in- and out- points while transferring DIF files.

+ Updates MSB clip cache so the clip is available for playout before the end of the transfer.

+ Does not quit on end-of-file for growing clips.

+ Can be used to set start, in and out points of a clip.

+ Logging (at various levels) in */usr/vtr/adm/logs/vtrftpdlog*.

vtrftpd accepts the options supported by **ftpd** (see FTPD(1M)). In addition, the following options may be specified:

-G*bps*
: Specifies the default I/O rate to be used when transferring files to or from a real-time subvolume. This value is used only if the file does not match one of the patterns in the configuration file or if the matching pattern does not specify an I/O rate. This option overrides **vtr.ftp.io.rate.default** as specified in the configuration file.

-P*pri*
: The daemon runs at the specified real-time priority, ensuring that transfers are not delayed by other activities (real-time or not) in the system. This option overrides **vtr.ftp.cpu.priority** as specified in the configuration file.

CONFIGURATION FILE

The real-time configuration file, **/usr/vtr/config/vtrftpd.conf**, can be used to specify the guaranteed disk bandwidth rate reserved by new files stored on the real-time subvolume of a filesystem.

Each line in the configuration file specifies a filename pattern and, optionally, a Guaranteed I/O bit-rate. For example:

```
        /usr/vtr/clips/new/* 10000000
```



```
        /usr/vtr/clips/*.dif 30000000
```

specify that all new files stored under /usr/vtr/clips/new will be transferred at a guaranteed disk bandwidth rate of 10Mbps, while all new files under /usr/vtr/clips with the file type ".dif" will be transferred with a disk rate of 30Mbps.

If the configuration file is missing or does not contain any filename patterns, new files will be stored into the real-time subvolume of a filesystem using a default I/O rate of 64000000. The default I/O rate may be specified using either the **-G** option or **vtr.ftp.io.rate.default** configuration variable.

The configuration file may specify values for certain configuration variables. The name of the variable is specified followed by whitespace and the value of the variable on a single line.

The following variables may be specified:

vtr.ftp.cpu.priority specifies the real-time priority that **vtrftpd** will run at during a file transfer. Real-time priorities are in the range 0 to 255, however the specified priority should not exceed 126.

vtr.ftp.io.rate.default specifies the default I/O rate to use for transfers to or from a real-time subvolume. This rate is used if no filename pattern matching the name of the file to be transferred is found in the configuration file.

vtr.ftp.io.rate.enforce specifies whether a transfer will attempt to use additional disk bandwidth beyond the Guarantee I/O rate specified for the transfer. If the value is 0, additional bandwidth may be used if available. If the value is 1, the transfer will not exceed the specified I/O rate. The default is to 1.

vtr.ftp.procs.max limits the maximum number of simultaneous transfers to the value of this control. The default is 100.

vtr.ftp.log_level.file specifies the level of logging that is directed to the log file (*/usr/vtr/adm/logs/vtrftpdlog*). A value of -1 logs only warnings and errors, 0 logs additional informational messages, and 1-3 correspond to debugging output. The default is 0.

vtr.ftp.marks disables mark points for DIF clips.

vtr.ftp.update.interval is the interval between successive MSB updates. A value of 0 never updates MSB's clip cache. A value of 1 will update the clip cache every time a block of data is transferred, a value of 2 will update every other time,etc.

vtr.ftp.update.failures is the maximum number of update failures before **vtrftpd** quits trying to update the clip cache. A value of 0 will disable updates.

vtr.ftp.datatype.default is the default datatype. 1 is TYPE_A (ascii), 2 is TYPE_E (ebcdic), 3 is TYPE_I (image/binary), 4 is TYPE_L (local).

See /usr/include/arpa/ftp.h The default is TYPE_A.

The following variables are intended for expert users:

vtr.ftp.timeout.flush is the time after which a receiving **vtrftpd** process flushes partial extents in memory (normally **vtrftpd** would wait for the full extent to be available before writing it out). The default is 100 msec.

vtr.ftp.timeout.transfer is the time after which a receiving **vtrftpd** process concludes that there is nothing more to read (the sending host could be dead or the link could be down), and ends the transfer instead of waiting indefinitely. The default is 60000 msec.

vtr.ftp.timeout.growth is the age in milliseconds after the last mod ofa file before **vtrftpd** decides that a file has stopped growing. The default is 10000 msec.

vtr.ftp.wait.growth is the sleep interval in clock ticks between successive checks for file growth. The default is 100.

vtr.ftp.wait.flush is the sleep interval in clock ticks before the last extent of a file is read. The default is 100.

MARKS

If a DIF file is ftp'd from the video server, and start, in and/or out points have been set on the DIF file, **vtrftpd** will transfer only the frames between the in and out points instead of the whole file. The whole file is transferred if the mark points are inconsistent or absent. To disable marks, the toggle command **site marks** may be used. The ability to transfer between mark points is limited to DIF clips only at this time.

UPDATES

vtrftpd attempts to update the clip cache by using the MVCP CADD command. This is useful in the absence of fsmon to keep the clip cache up-to-date. The frequency of updates and the tolerance to failures may be controlled using the above configuration controls.

STREAMING

vtrftpd is able to stream a clip that is being recorded before the recording is finished. When **vtrftpd** reaches the end of file, it does not immediately terminate the connection. Instead it polls for file growth. The above controls can be used to control the sleep periods in between checks or the length of time that must elapse after the last modification to a clip before **vtrftpd** concludes that the file is no longer growing. Please note that in order to have this tail mode facility for index files, you must use the "site win" command. You can

also daisy chain a number of ftp's to stream a file through a string of servers.

SITE COMMANDS

The following MSB-specific site commands exist:

site marks toggles between enabling and disabling mark points. The

default is that marks are enabled. This feature is relevant only to DIF clips with valid mark points.

site editpoints clip prints the editpoints of clip *clip*.

site editpoints tc1 tc2 tc3 clip sets start, in and out points of *clip* to *tc1, tc2, tc3* respectively.

LIMITATIONS

vtrftpd cannot be used to transfer MSB content of format movie/vframe or movie/vclip. It can be used to transfer mpeg2 files, but it is up to the application to (re-)generate MSB mpeg2 index files via the **vtrmpegutil** utility or transfer the index files as well. It does not respect in/out points on any format other than DIF. An attempt to use **vtrftpd** to transfer a clip of one of the above unsupported formats may appear to work, but the clip will not be usable.

SEE ALSO

ftpd(1M), *vtrmpegutil*(1M)

vtrhwinfo(1)

NAME

vtrhwinfo - Retrieve MSB hardware manifest

SYNOPSIS

/usr/vtr/bin/vtrhwinfo [*options*]

DESCRIPTION

vtrhwinfo retrieves MSB hardware manifest. An inventory of installed hardware items, such as PCI-VIDAUD-MSB codecs, are reported. This is an internal command, used by VST to acquire a list of installed peripherals controlled by VST. Application of this command for use other than VST internal purposes is not supported.

SEE ALSO

vtrswinfo(1)

vtrstart(1)

NAME

vtrstart - MSB startup

SYNOPSIS

/usr/vtr/bin/vtrstart

DESCRIPTION

vtrstart is the system startup script for the MSBplatform software. To start the MSB, simply run the **vtrstart** script.

MSB runs as a background daemon, so there will be no output to indicate that it is running. You may use **vtrstat** to verify the status of the MSB primary server after you have started it.

vtrstart starts the MSB parent daemon, **vtrd**, using site-dependent options and arguments in the file */usr/vtr/config/vtrd.options*.

SEE ALSO

vtrstat(1), *vtrd*(1), *vtrstop*(1)

vtrstat(1)

NAME

vtrstat - MSB status

SYNOPSIS

/usr/vtr/bin/vtrstat [**-h** *hostname*]

DESCRIPTION

vtrstat is the MSB status display utility. It can be used to check the status of the MSB system.

vtrstat has the following options:

-host *hostname*
Check the status of MSB on the host specified by *hostname.* If this option is not specified, **vtrstat** will check the local host.

-ports
Shows a list of the signal ports supported by the MSB.

-units
Shows a list of the units currently open on the MSB.

-unitfunction *function*
Shows only those units which match the specified *function*. Also specify **-unitstatus run** to show only those units that are currently executing *function*.

-unitstatus *status*
Shows only those units which match the specified *status*.

-statvalues *value-pattern*
Display a table of statistics values which have names that match the specified *value-pattern*.

-statinstances *instance-pattern*
Display a table of statistics values which have instances names that match the specified *instance-pattern*.

SEE ALSO

vtrstart(1), *vtrstop*(1)

vtrstop(1)

NAME

vtrstop - MSB shutdown

SYNOPSIS

/usr/vtr/bin/vtrstop

DESCRIPTION

vtrstop is the system shutdown script for the MSBplatform software. To stop the MSB parent daemon, simply run the **vtrstop** script.

MSB runs as a background daemon, so there will be no output to indicate that it has been stopped. You may use **vtrstat** to verify the status of the MSB server after you have stopped it.

SEE ALSO

vtrstat(1), *vtrstart*(1)

vtrerrinfo(1)

NAME

vtrswinfo - Retrieve MSB port status

SYNOPSIS

/usr/vtr/bin/vtrerrinfo

DESCRIPTION

vtrerrinfo retrieves codec port status for the MSB. Summary status information such as dropped frames, video underflow, port crashes, loss of synchronization or signal input, are reported.

vtrerrinfo has the following options:

-host *MSB hostname to report on*
-allports *report error information for all codecs and ports*; *report on first port otherwise.*

SEE ALSO

vtrhwinfo(1)

vtrswinfo(1)

NAME

vtrswinfo - Retrieve MSB software manifest

SYNOPSIS

/usr/vtr/bin/vtrswinfo

DESCRIPTION

vtrswinfo retrieves the MSB software manifest by processing the **versions(1M)** command output. An inventory of installed software items, such as *vst_eoe.sw*, are reported. This is an internal command, used by VST to acquire a list of installed software bundles controlled by VST. Application of this command for use other than VST internal purposes is not supported.

SEE ALSO

vtrhwinfo(1)

vtrsyncinfo(1)

NAME

vtrsyncinfo - MSB video sync information tool

SYNOPSIS

/usr/vtr/bin/vtrsyncinfo [**-host** hostname]

DESCRIPTION

vtrsyncinfo is a video synchronization information utility for MSB.It can be used to determine if a system's video input and output portsare in-sync.

vtrsyncinfo has the following options:

-host *hostname*
Check the video sync of the video ports on the host specified by *hostname*. If this option is not specified, **vtrsyncinfo** will check the local host.

-baseport
Specify the video port against which other ports will be compared in determining their sync phase differences. If this option is not specified, **vtrsyncinfo** will use the first video port found. A video port is specified in the same manner as it is displayed by this tool: "MFCODEC_2.DigitalInput" or "MFCODEC_3.DigitalOutput". For example, to specify the input port of MFCODEC_2, use "MFCODEC_2.DigitalInput". (NOTE: a house timecode port cannot be specified as the baseport)

-probe
Cause **vtrsyncinfo** to attempt to start all video ports in the system before sampling their video sync. Choosing this option will cause color bars to be output by all output video ports not currently in use, so use with caution. If this option is not specified, sync information for any video ports that are not running will be missing or inaccurate.

-tc
Cause **vtrsyncinfo** to display sync phase information for any configured house timecode inputs in the system.

-loop
Cause **vtrsyncinfo** to re-sample and re-display sync information once per second.

-help
Display **vtrsyncinfo** usage information

-verbosity *level*
Specify *level* greater than 0 for debugging information.

DISPLAYED INFORMATION

vtrsyncinfo displays three pieces of information for each port:

video standard
Indicates if the port is synced to a 525-line or 625-line signal. For house-timecode ports, this also indicates drop-frame or non-drop-frame mode in 525 mode.

sync source
Indicates the video sync signal to which the port is synced. For input ports, this is always the input signal. For output ports, it will be one of the following:

internal	Internal clock
external	External genlock (GEN-IN) video ref signal
digital-input-link-a	External genlock to digital input link A
digital-input-link-b	External genlock to digital input link B

phase offset
Indicates the vertical and horizontal phase offset of this port with respect to the base port. Vertical phase offset is displayed in lines. Horizontal phase offset is displayed in microseconds.

SEE ALSO

vst(1), *vtrstat*(1), *vtrswinfo*(1), *vtrhwinfo*(1)

vtrvfutil(1)

NAME

vtrvfutil - MSB vframe clip utility

SYNOPSIS

/usr/vtr/bin/vtrvfutil [options ...] *clip* [*new-clip*]

DESCRIPTION

vtrvfutil is a utility for examining and modifying VCP-Recorder clips stored using the *vframe* clip format. *clip* specifies the name of the clip: Not the full filename, as in /usr/vtr/clips/d/news, but rather the clip name, d/news.

If *new-clip* is specified, *clip* is copied to *new-clip*. The clip alignment can be changed by using the **-A** options.

Each **vtrvfutil** option may have one or more suboptions that are specified using the format:

-x subopt1[=value1],...,suboptn[=valuen])

Some suboptions take values, while others do not.

vtrvfutil has the following options:

Audio options:

-a export=*file*
Export audio to *file* (AIFF-C format)

-a import=*file*
Import audio from *file* (AIFF-C format)

-a in=*h:m:s:f*
Timecode of in point (starting frame)

-a out=*h:m:s:f*
Timecode of out point (ending frame +1)

Alignment options:

-A major=*bytes*
Specify clip major (stripe) alignment

-A minor=*bytes*
Specify clip minor (block) alignment

Command options:

-c checknondrop
Check for non drop vitc timecode

-c checkrice
Software decodes each field of Rice-coded clip

-c checkvitc
Check for vitc discontinunity

-c copy
Copy input clip to output clip

-c makeindex
Make vframe index (input clip in Quicktime format)

-c makeqt
Make Quicktime (add Quicktime metadata to clip file)

-c realign
Realign clip in place (use -A to specify alignment)

Input clip options:

-i file=*file*
Specify clip filename

-i index=*file*
Specify clip index filename

-i nodirect
Do not use direct I/O to access clip

-i qt
Clip is in Quicktime format (no vframe index)

Output clip options:

-o file=*file*
Specify clip filename

-o index=*file*
Specify clip index filename

-o nodirect
Do not use direct I/O to access clip

-o vitc
Use vitc as frame count to generate index file

Printing options:

-p headers
Print all clip headers

-p info
Print clip header summary and analysis

-p frames
Print clip per-frame info

-p sizes=*bins*
Print frame-size histogram

Still image import/export options:

-s export
Copy the image at the specified timecode from the clip to the specified file. This is the default when **-s** is specified but neither **import** nor **export** is specified.

-s file=*file*
Specify image file name

-s import
Copy the image from the file into the clip at the specified timecode.

-s mode=*mode*
Specify interleave mode (f1, f2, f1f1, f2f2, f1f2)

-s type=*type*
Specify image format (rice, yuv, rgb, jpeg, tiff)

-s tc=*h:m:s:f*
Specify clip frame to import or export

Update options:
NOTE: If a new-clip name is specified, the update is applied to *both* the original clip and the new clip.

-u header
Update vframe index header using analysis results

-u frames=*num*
Set number of frames to *num*

-u start=*h:m:s:f*
Set starting frame timecode

-v *level*
Set verbosity level

QUICKTIME CONVERSION NOTES

The SGI MovieLib support for Quicktime-formatted files only supports audio sample widths of 8 or 16 bits. This may conflict with the default input audio sample width for the device port you are using to record your clips.

EXAMPLES

To copy clip A to clip B:

```
vtrvfutil A B
```

To copy clip A to /clips/B.{media,index}:

```
vtrvfutil -o file=/clips/B.media,index=/clips/B.index A B
```

vvtr(1)

NAME

vvtr - MSB process

SYNOPSIS

/usr/vtr/bin/vvtr [**-B** *bytes*] [**-f** *log-level*]
[**-F** *log-file*] [**-l** *log-options*] [**-p**] [**-P** *priority*]
[**-s** *log-level*] [**-u** *num-units*] [**-v** *log-level*]

DESCRIPTION

The MSB process server is the main executable of the MSBplatform software. Note, however, that under normal circumstances vvtr is not run on its own but is launched by the MSB parent daemon, **vtrd**.

vvtr has the following options:

-B *bytes*
Set the default stream I/O buffer size to *bytes*.

-f *log-level*
Set the maximum log message priority for the log file to **Info**+*log-level.* The default is 0, meaning all log messages up to and including **Info** priority are written to the log file.

-F *log-file*
Write the log file to *log-file*. The default is */var/adm/vtr/logs/vtrlog*.

-l *log-options*
Omit specified log message fields from log messages. *log-options* is one or more of the following:

l
Omit priority level indicator.

p
Omit process id (pid).

t
Omit timestamp.

Level, process id, and timestamp are always omitted from log messages in SYSLOG.

-p
Do not use real-time scheduling priorities.

-P *priority*
Set base scheduling priority to *priority*. The default is system-dependent.

-s *log-level*
Set the maximum log message priority for SYSLOG to **Info**+*log-level.* The default is 0, meaning all log messages up to and including **Info** priority are written to SYSLOG.

-t

Truncate the log file when it is opened. If this option is not specified, the new log messages are appended to the existing log file.

-u *num-units*
Set the maximum number of concurrent units to *num-units*. This reserves statically-allocated resources to support the specified number of units. The default is 32.

-v *log-level*
Set the maximum log message priority for stdout to **Info**+*log-level.* The default is 0, meaning all log messages up to and including **Info** priority are written to stdout.

LOGGING

MSB logs to several destinations. The **-f**, **-s**, and **-v** options are used to adjust the lower bound on the priority of messages logged to each of the log file, system log, and stdout.

The **-F** option can be used to change the name of the log file (the default is /usr/vtr/logs/vtrlog). If the **-t** option is used, the log file is truncated each time it is opened; otherwise, the new log messages are appended to the existing log file.

If MSB is sent a SIGHUP signal, it responds by reopening the log file. SIGHUP can be used to rotate log files while MSB remains running.

SEE ALSO

vcp-recorder(1), *vtrd*(1)