The SGI Media Server for Production and Broadcast - DVCPRO

Document Number 007-3885-001

Written by Ken Jones

Illustrated by Chris Wengelski

Production by Susan Gorski

Engineering contributions by C. Jason Mancebo, Kurt Merriweather

Cover design by Sarah Bollas, Sarah Bollas Design, and Dany Galgani, SGI Technical Publications..

Record of Revision

Version	Description
001	May 2000 Intial publication

Contents

Figures

Tables

About This Guide

The SGI Media Server for production and broadcast is a completely integrated solution for serving video, audio, and data that is based on open architecture, data networking, and storage technology. At the heart of the multichannel media server is SGI's Video Server Technology (VST) software. The current product supports the DVCPRO news format.

The manual describes how to use the media server to play and record digital media and to store the data in, and retrieve it from, a StudioCentral 2.0 archive system.

Also described in this document are the graphical user interfaces (GUIs), which are used to manually control the media server, and MVCP (Multiport Video Computer Protocol), which is a command-line, control protocol supported by VST.

What This Document Contains

The following material is covered in this document:

- Chapter 1, "Overview of the SGI Media Server," provides a functional overview of the SGI Media Server and then describes its software and hardware.
- Chapter 2, "System Setup," describes the general setup requirements of your media server.
- Chapter 3, "System Configuration,"describes the various configuration requirements and the mechanisms for configuring and controlling device-independent and device-specific features.
- Chapter 4, "Running the SGI Media Server," describes starting, monitoring, and stopping the Video Server Technology (VST) software.
- Chapter 5, "Using the SGI Media Server Control GUI," describes the graphical user interface (GUI) that you use to control and status system activities like playing and recording clips.

- Chapter 6, "Adding and Removing Clips," describes how you add and remove clips from the media server.
- Chapter 7, "Using Clip Manager," describes how to use the Clip Manager to manipulate clips and their attributes.
- Chapter 8, "Virtual Clips," describes virtual clips and how you manipulate them on the media server.
- Chapter 9, "Configuring and Using External Devices," describes how to handle the external devices connected to the media server.
- Chapter 10, "Troubleshooting," describes some likely problem scenarios you might experience while operating your media server and what to do.
- Chapter 11, "Completing Common Tasks Using MVCP Commands," describes a grab bag of common tasks you routinely perform using MVCP commands.
- Appendix A, "Setting Up the Hardware," describes the steps you need to take to set up your system so that Video Server Technology (VST) can run on it.
- Appendix B, "Setting Up Filesystems for the SGI Media Server," describes how to set up XFS filesystems for the SGI Media Server and for the clip cache.
- Appendix C, "Installing Video Server Technology," provides a step-by-step description of installing the core software.
- Appendix D, "IRIX Monitoring Tools," describes a collection of IRIX monitoring tools that can be used with the media server.
- Appendix E, "4:2:2:4 Sampled Video," describes the recording and playout of 4:2:2:4 sampled video with alpha.
- Appendix F, "Configuring a Server for Clip Mirroring," describes how you can set up one or more servers for clip mirroring with clip caches that mirror the clip cache on a designated primary server.
- Appendix G, "Archiving Clips," describes the archiving of clips to an archival server.
- Appendix H, "Multiport Video Computer Protocol (MVCP) Command Summary," describes the MVCP commands.
- Appendix I, "RS-422 Pinouts," describes the RS-422 pinouts used with the SGI Media Server.

The glossary provides definitions of key words used in this document.

An HTML version of this book is installed at URL:

```
http://hostname.domain/MS/MS400_800_UG.html
```

Who Should Read This Document

This document is written for SGI Media Server user , system integrators, and others who are interested in obtaining an overview of the product. It is assumed that the reader is already familiar with broadcast industry concepts.

Related Documentation

Refer to the man pages for specific command help. The man page titles are:

- vtrstart(1)— for startup
- vtrstop(1)— for shutdown
- vtrstat(1)— for status
- vtrclip(1)— for clip management
- mcpanel(1)— for media control panel
- mcclips(1)— for clip manager
- mcstat(1)— for status display
- mccompstats(1)— for compression monitor
- vcp-recorder-controls(5)— for VST controls
- mvcp(5)— Multiport Video Computer Protocol
- vvtr(1)— for VST server
- vtrd(1)— for VST daemon
- vtrvfutil(1)— for VST vframe clip utility

You can list the man pages by entering the following command:

```
% versions long vcp_eoe | grep man
```

Refer to the following documents for supplementary information:

- *IRIX Admin: Software Installation and Licensing* (part number 007-1364-*nnn*) for information about installing software that runs under IRIX, the SGI implementation of the UNIX operating system
- *IRIX Admin: System Configuration and Operation* (part number 007-2859-*nnn*) for information about IRIX system administration tasks
- *IRIX Admin: Disks and Filesystems* (part number 007-2825-*nnn*) for information about general filesystem concepts and system administration procedures for SCSI disks, XFS and EFS filesystems, logical volumes, and guaranteed rate I/O.

Conventions Used in This Document

The following type and symbol conventions are used in this document:

Italics	Used for filenames, pathnames, directory names, emphasis, document titles, variable names, glossary terms, and command-line programs
Bold	Used for keywords
`Fixed-width`	Used for code examples and command syntax
`Bold fixed-width`	Used for user input, including nonprinting keyboard keys
Square brackets ([])	Surround syntax statement arguments that are optional
Square bullets	Indicate substeps within a multistep process
Ellipsis (...)	Indicate that the preceding is repeated
Right angle brackets (>)	Indicate a path through menus to a menu option. For example, “File > Open” means “Under the File menu, choose the Open option.” Right angle brackets also indicate the play button in the graphical user interface.

Reader Comments

If you have comments about the technical accuracy, content, or organization of this document, please tell us. Be sure to include the tittle and part number of the document with your comments.

You can contact us in any of the following ways:

- Send e-mail to the following address:

  `techpubs@sgi.com`

- Send a fax to the attention of “Technical Publications” at: +1 650 932 0801.

- Use the Feedback option on the Technical Publications Library World Wide Web page:

  `http://techpubs.sgi.com`

- Call the Technical Publications Group, through the Technical Assistance Center, at: 1 800 800 4SGI.

- Send mail to the following address:

  Technical Publications
  SGI 1600 Amphitheatre Pkwy.
  Mountain View, California 94043-1351

We value your comments and will respond to them promptly.

Overview of the SGI Media Server

This chapter describes the SGI Media Server, which provides real-time, frame-accurate recording and playback of broadcast-quality digital media data. The media server manages video as data, distributing the files over the existing LAN / WAN infrastructure--whether Fibre Channel, 100Base-TX, Gigabit Ethernet, or enhanced video networks--within a facility or between facilities.

The following topics are discussed in this chapter:

- "Functional Overview" on page 1
- "Software Overview" on page 4
- "Hardware Overview" on page 12

Functional Overview

Digital media data is brought into the media server by recording it from a live feed or a videotape deck, retrieving it from a StudioCentral 2.0 archive system, or copying it from a file. The data can then be played out to a broadcast system, a video port, or a videotape deck or transferred to a StudioCentral 2.0 archive system for storage and distribution.

The media server can be automatically controlled by an application or through the use of a broadcast system *automation controller,* which can control video servers using serial control protocols, such as the Louth VDCP protocol, or an edit controller that can control a VTR using the Sony RS-422 VTR protocol.

The media server can also be manually controlled by using the SGI Media Server graphical user interface (GUI) or can be controlled by an application using the Multiport Video Computer Protocol (MVCP). The GUI consists of screens that are used to record and play digital data, determine status information, and manage digital media data stored on the media server.

This functionality is shown in Figure 1-1.

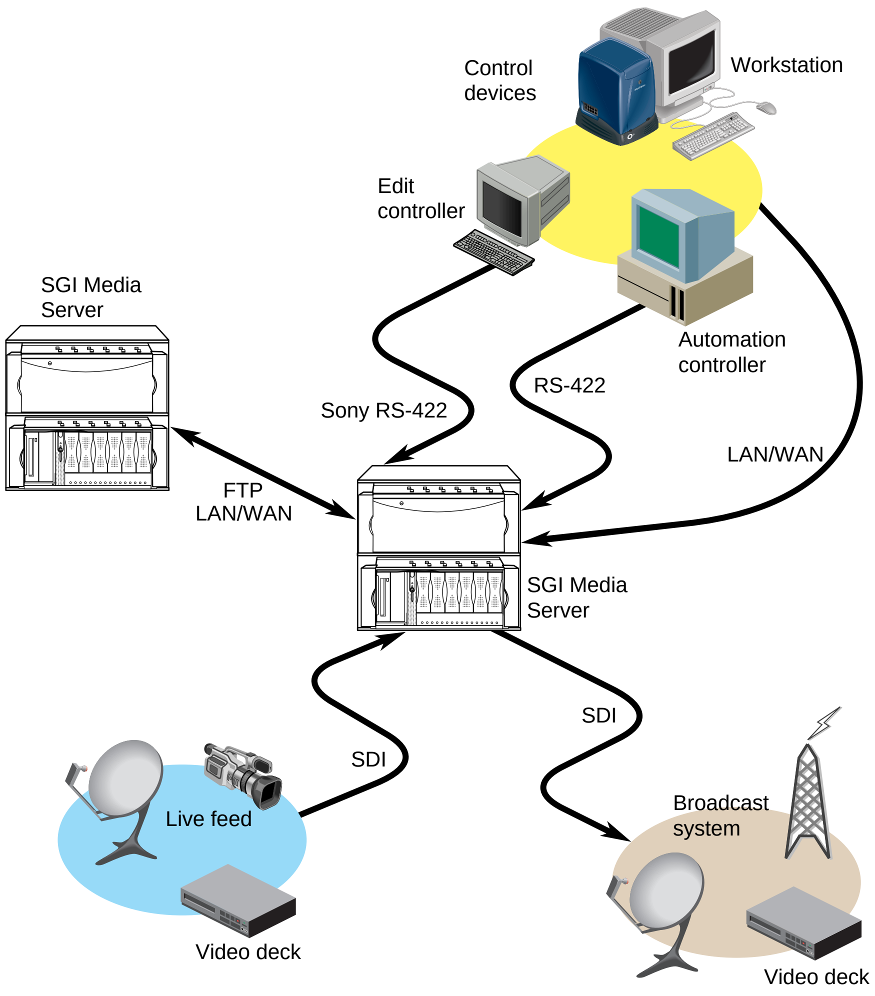


Figure 1-1 Functional Overview

The GUI screen that is used to record and play digital media data is the Media Server Control Panel (mcpanel), which is shown in Figure 1-2. The control panel is similar in function to a standard videotape player or recorder. For example, there are buttons in the control panel to cue the video, play it, stop the playback, and so on.

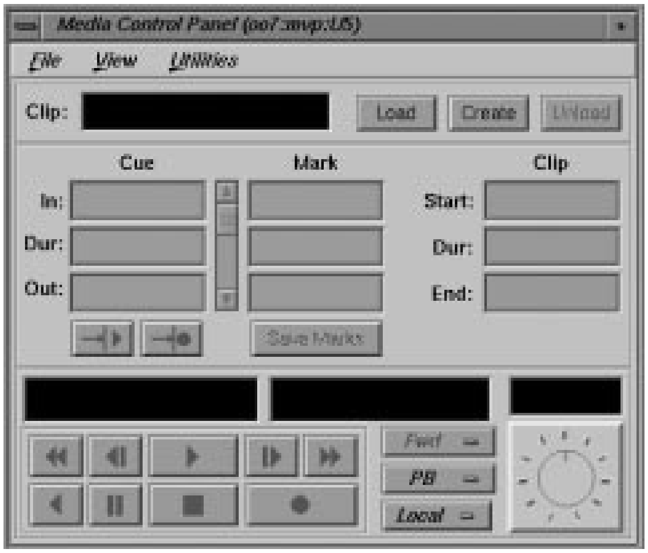


Figure 1-2 Media Server Control Panel

The graphical user interface is described in Chapter 5, “Using the SGI Media Server Control GUI.” The MVCP protocol, which was used to implement the GUI, is described in “Protocol Format” on page 233.

Software Overview

The SGI Media Server software provides scalability and maximum flexibility. The software includes the following:

- Video Server Technology (VST), the core software of the media server. VST provides real-time, frame-accurate recording and playback of broadcast-quality digital media.
- Control interface modules, which provide device-dependent code. For example, there is a control interface module that contains the code that is specific to a Louth *automation controller*.
- Media device interface modules, which contain format-dependent code that provides access to the *port*s over which media is played and recorded.
- Format interface modules, which provide handlers for accessing specific digital media storage formats. For example, there is a format interface module for the *DVCPRO* Data Interchange Format (DIF).

Figure 1-3 shows the primary software components in VST.

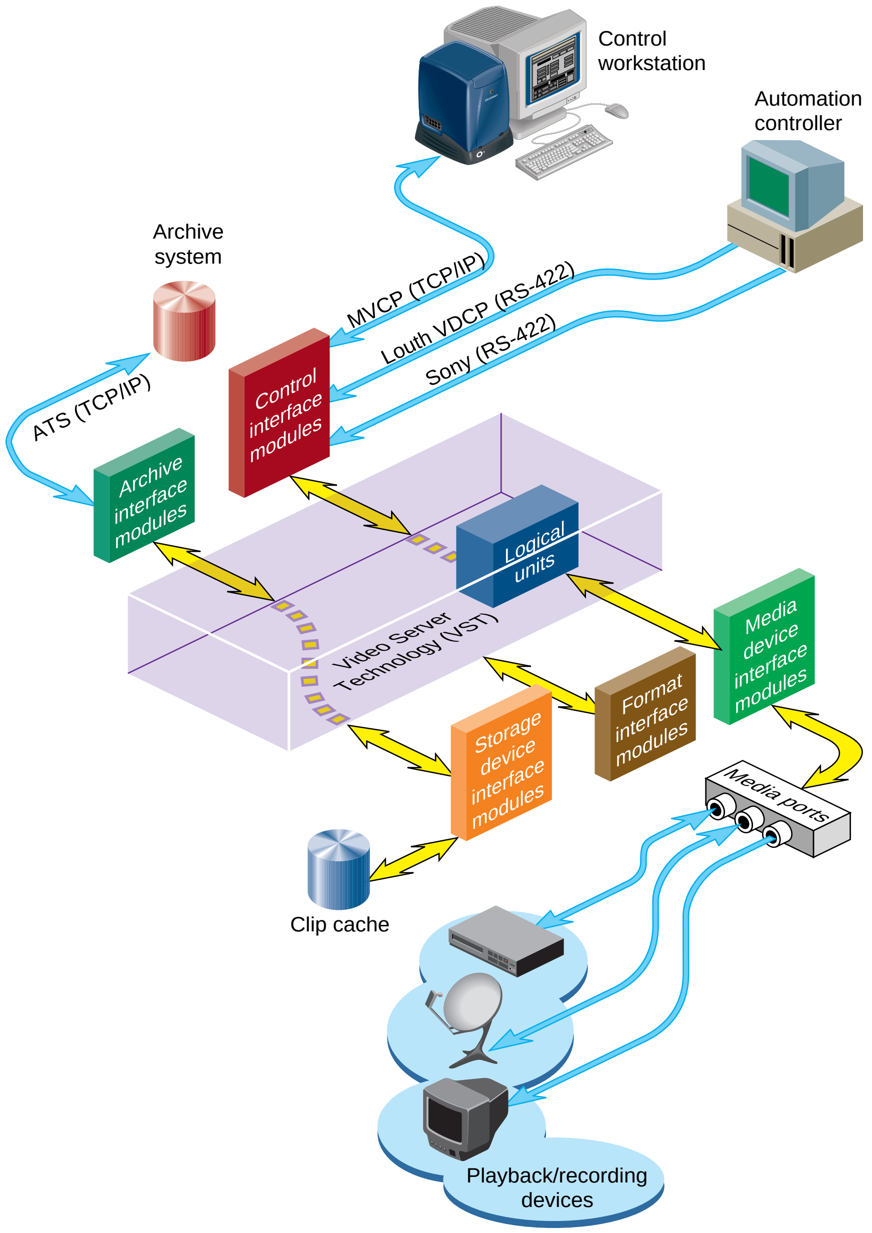


Figure 1-3 Video Server Technology Software Components

The remainder of this section discusses the software components shown in Figure 1-3.

Clip Cache

Digital media data that media server processes for playout and recording is stored in one or more *clip cache*s. Each unit of data that is stored in a clip cache (for example, a movie) is called a *clip*.

Clips can be added to the cache by doing the following:

- Using the media server to record the clip
- Generating the clips elsewhere and adding them to the cache

Clips can be transferred from the clip cache into a StudioCentral 2.0 archive system for storage.

Core Software

The VST core software provides the basic functionality for playback and recording of media. It utilizes the IRIX operating system as well as portions of the SGI Digital Media Libraries.

The core software provides the following basic functionality:

- Clip cache management, which maintains persistent information about the media that is either stored in the *clip cache* or is in the process of being transferred into or out of it
- Controller management, which links one or more external control protocol modules (for example, *Louth*) to the internal processing logic
- Configuration management, which automatically configures the VST software according to the hardware capabilities of the system on which it runs

VST provides a core library that supports external interface modules, dynamic shared objects (DSOs) that contain the code specific to a given external entity. When the media server is started, the VST software loads and initializes all external interface modules it locates so that the modules can be used.

Control Interface Module

Control interface modules allow various *automation controller*s and digital media applications to control the use of the SGI Media Server. These modules translate to and from external control protocols.

The following control interface modules are provided:

- The Louth Video Disk Communications Protocol defined by Louth Automation. This control protocol provides control of VST over RS-232, RS-422, or TCP/IP. The VST's Louth interface module supports back-to-back play and record (subject to restrictions imposed by the video I/O port capabilities) and archive management.
- The Sony RS-422 VTR (also called, 9-pin or P2) protocol. VST supports this protocol through a full-featured VTR deck-emulation mode that includes frame-accurate insert editing and variable-speed shuttle.
- Multiple-Unit Video Computer Protocol (MVCP) defined by SGI. This control protocol provides full-featured control of VST through TCP/IP. This control interface module supports archive management, multiple-unit control, and event monitoring, and provides access to advanced features of SGI devices.

Storage Device Interface Module

Storage device interface modules provide access to the storage systems on which the *clip cache* resides. Currently, there is a storage device interface module for the IRIX XFS filesystem.

Media Device Interface Module

Media device interface modules provide access to the *ports* over which the media is played and recorded (that is, the media ports). Each type of I/O port typically has its own media device interface module.

VST has media device interface modules for the following:

- DIVO-DVC is the DVCPRO version of the DIVO card. It performs all the functions of the DIVO card and DVCPRO.
- Diaquest. Direct Sony 422 control of videotape decks; used in the same way as V-LAN.

DVCPRO Format

The DVCPRO compression algorithm compresses four video frames into one frame. This compression format is useful for transporting video data across networks, such as between video decks. DVCPRO-compressed frames in 1 times mode can be displayed normally. The following section describes how to display DVCPRO-compressed frames in 4 times mode

Displaying 4 Times Mode DVCPRO-Compressed Frames

The Serial Digital Transport Interface (SDTI) is a video networking protocol that allows arbitrary data to be packeted and transmitted over the SMPTE-259M Serial Digital Interface (SDI). VST supports playback and recording of DVCPRO (DIF) over SDTI at 1 times and 4 times normal speed. When recording SDTI / DVCPRO using DIVO-DVC, the incoming signal can be looped to the output (EE mode) in either 1x or 4x mode.

Logical Playback and Record Units

Logical *unit*s enable media *port*s to play and record *clip*s. Each the media server unit can be thought of as a logical videotape recorder (*VTR*) transport that is capable of loading, cueing, playing, and recording clips using a specific media port.

Logical units are created automatically by VST when the SGI Media Server Control GUI or an *automation controller* is used. When the *MVCP* protocol is used, a command requests that a unit be created or that a unit created by another control connection be used.

There is normally a one-to-one relationship between a control connection to the media server and a VST logical unit, and between a logical unit and a media port. This is shown in Figure 1-4.

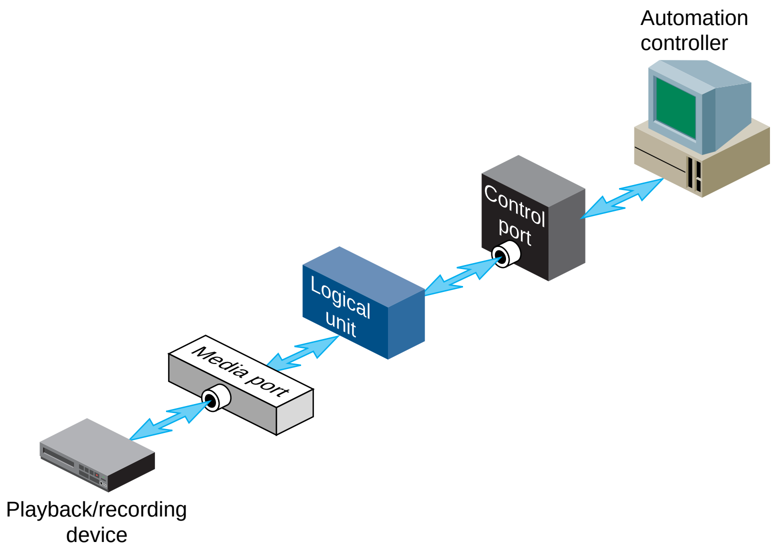


Figure 1-4 One Logical Unit With One Control Connection

A single unit can also be controlled by multiple control ports. For example, two tightly integrated applications might control a single unit, where each application would have its own control port. This is shown in Figure 1-5.

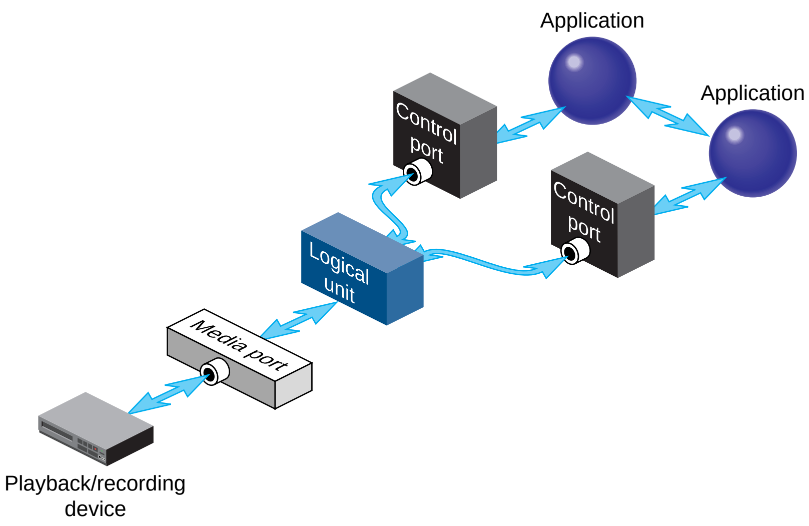


Figure 1-5 One Logical Unit With Two Control Connections

Caution: Exercise extreme care when the control of a unit is shared between two controller connections. Interfering with a unit owned by a Sony or Louth *automation controller* leads to unpredictable behavior.

A media port can be controlled by multiple logical units. For example, an application with one control connection and two units could be cueing one clip while playing out another, enabling back-to-back playout of clips when allowed by the media format. This example is shown in Figure 1-6.

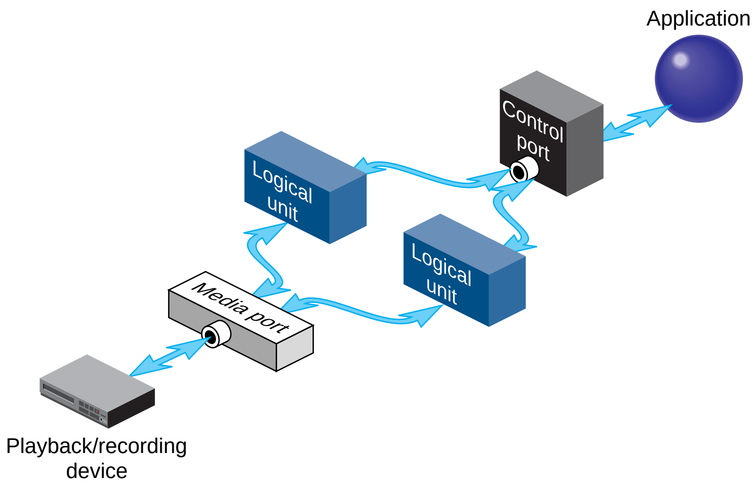


Figure 1-6 Two Logical Units With One Control Connection

If a media port supports multiple logical units, the sharing is subject to the device-sharing characteristics of that port.

Hardware Overview

The SGI Media Server hardware configuration consists of the following components:

Base Unit	Origin200 with GIGAchannel
Storage	Internal ultraSCSIs or external CIPRICO 7000 Fibre Channel RAID-3 Arrays
Internal Devices	Video I/O: DVCPRO-25 Audio I/O: Digital Audio I/O (AES/EBU or ADAT) Data Networking: 100 Base-TX, 1000 Base-SX, FibreChannel
External Devices	Devices using Silicon Graphics Multi-Unit Video Computer Protocol(MVCP TCP/IP) Panasonic AG-A850 editing controller (Sony/P2 RS-422 protocol) Buf VTC-4000 multi-VTR controller (with or without Bufnet/RS-422adapter) Buf RM-4000 remote processor (Sony/P2 RS-422 protocol) Louth automation controller (Louth RS-422 protocol)

A typical hardware configuration is shown in Figure 1-7.

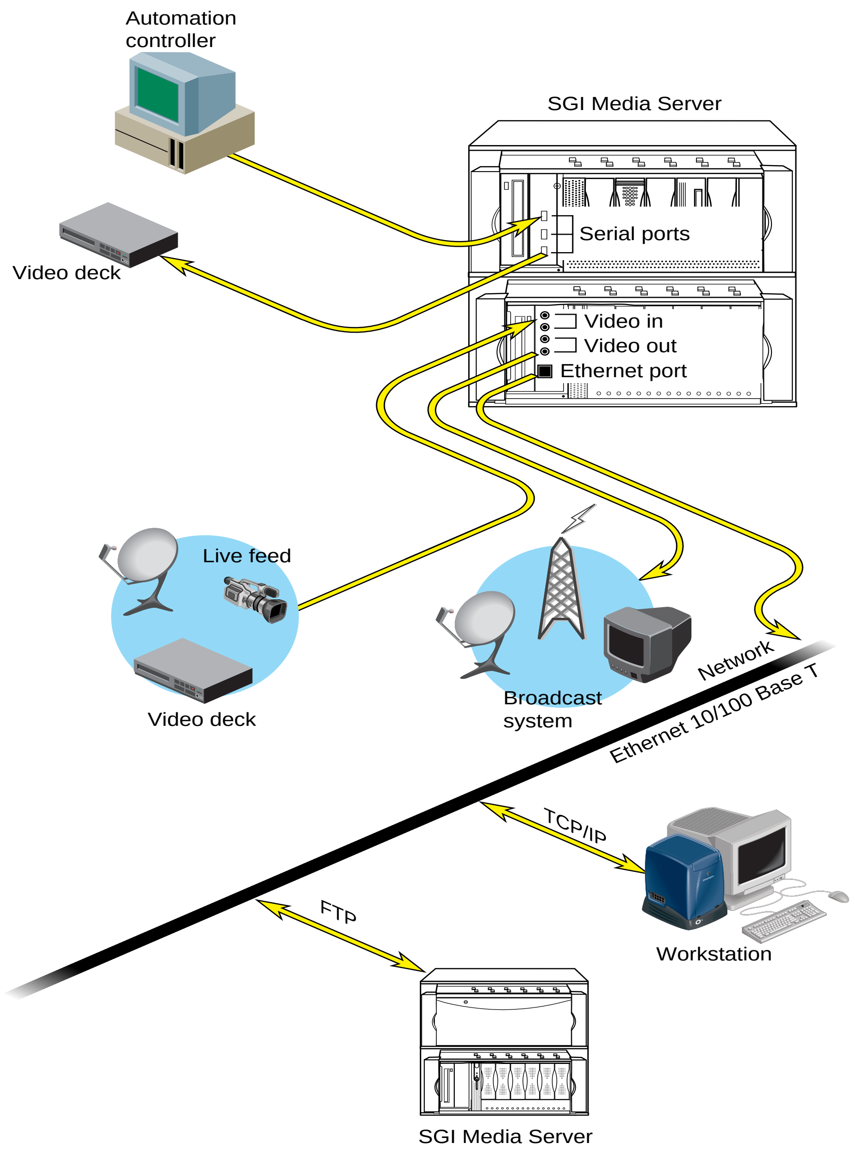


Figure 1-7 SGI Media Server Hardware Components

The Origin200 Base Unit

The Origin200 with GIGAchannel provides massive processing, storage, and throughput capabilities to satisfy even the largest production requirements. They are built from a scalable node architecture, enabling small configurations that can be incrementally upgraded to the larger configurations. Each Origin200 server can be configured as a single module or as multiple modules with a single system image.

The Origin200 provides the following:

- DVCPRO, Data Interchange Format (DIF), a compression algorithm that compresses four video frames into one frame. This compression format is useful for transporting video data across networks, such as between video decks.
- GIGAchannel expansion box for the Origin200, which allows the addition of up to five DIVO-DVC boards.

Disk Storage

The SGI Media Server disk storage holds the XFS real-time filesystems that contain the movies, trailers, commercials, and other digital media data stored in the *clip cache*. The descriptive information about *clip*s, for example, clip names, duration, *edit point*s, and so on, are also stored on these filesystems.

The storage system supports the use of scalable storage to enable the total disk space to range from only a few gigabytes to hundreds of terabytes or more. The type of disk storage that is used depends upon several factors, including the number and size of stored clips, the use of RAID, and the required availability (uptime) of the system.

Two different types of disk storage are available:

- Standard disk storage, sometimes called RAID-0, in which several disk drives are striped into XLV logical volumes. This type of disk storage does not provide redundancy but it does provide higher bandwidth than XFS on a single disk drive.
- RAID storage, such as RAID-3, which provides high-availability, redundant digital storage.

External Devices

The media server supports the use of the following external devices:

- Devices using Silicon Graphics Multi-Unit Video Computer Protocol (MVCP TCP/IP)
- Broadcast system *automation controller*s that use, among others, the Louth Video Disk Communications Protocol, defined by Louth Automation. Automation controllers are connected to the server through an RS-422 serial connection or a TCP/IP Ethernet connection.
- Edit controllers, including the following:
    - Panasonic AG-A850 editing controller (Sony/P2 RS-422 protocol)
    - Buf VTC-4000 multi-VTR controller (with or without Bufnet/RS-422 adapter) and Buf RM-4000 remote processor (Sony/P2 RS-422 protocol)

For more information about installing these devices for use with the media server, see Chapter 9, “Configuring and Using External Devices.”

For a complete list of external device that interface with the media server, see the release notes.

Internal Devices

The SGI Media Server supports the use of the following internal devices:

- Video I/O: DVCPRO-25, which compress four video frames into one frame using cosine compression.
- Audio I/O: Digital Audio (AES/EBU or ADAT optical), which provides two channels of AES/EBU audio input and output per video stream.
- Data Networking: 100 Base-TX, 1000 Base-SX, FibreChannel

Video Device Control

The SGI Media Server can also control remote video devices directly using Sony 422 deck-control software provided by Diaquest.

PART ONE

System Setup and Configuration I

This part of the book gives an overview of the SGI Media Server and explains how to set up your media server.

Chapter 2, “System Setup,”describes the general setup requirements of your media server.

Chapter 3, “System Configuration,” describes the various configuration requirements and the mechanisms for configuring and controlling device-independent and device-specific features.

System Setup

This chapter describes setting up your system in the following sections:

- "Power Requirements" on page 20
- "Space Requirements" on page 21
- "Video Connections" on page 22

For a more comprehensive description of installing the various system components, see Appendix A, "Setting Up the Hardware."

Power Requirements

Power requirements for the SGI Media Server are as follows in Table 2-1:

Table 2-1 Power Requirements for the SGI Media Server

Specification	Value for 4-Channel System	Value for 8-Channel System
Input volts	100 to 120 VAC and 200 to 240 VAC,autoranging	100 to 120 VAC and 200 to 240 VAC, autoranging
Input watts	1381 watts[a]	2762 watts[a]
Input amps	<16 amps at 100 VAC[a] <8 amps at 240 VAC	<32 amps at 100 VAC[a] <16 amps at 240 VAC
Input volt-amps	1600 VA[a]	3200 VA[a]
Inrush current	280 amps maximum[a]	560 amps maximum[a]
Frequency	47 Hz to 63 Hz	47 Hz to 63 Hz
Output volts	+5 VDC +12 VDC -12 VDC +3.45 VDC	+5 VDC +12 VDC -12 VDC +3.45 VDC
Efficiency	60%	60%

a. The 4-channel media server and the 8-channel media server are comprised of 2 and 4 base units, respectively. Please note that all values are the aggregation of multiple power supplies.

Table 2-2 shows the power and size specifications for external RAID storage systems:

Table 2-2 Power and Size Specifications for the SGI Media Server External RAID Storage System

Specification	Value
Input Volts	100 to 120 VAC, 50/60 Hz, 6 amps, autoranging 200 to 240 VAC, 50/60 Hz, 3 amps, autoranging
Dimensions	7” x 17” x 24” (height, width, depth)
Weight	Approximately 75 pounds

Space Requirements

The 4-channel media server requires 8 standard rack units and the 8-channel media server requires 16 rack units. The media server occupies a standard 19” rack. In addition, each external RAID storage device requires 4 rack units and resides in a standard 19” rack.

Video Connections

The SGI Media Server contains up to 8 channels of video I/O, employing one SGI DIVO-DVC I/O card. Connections to the card are standard BNC connections for SDI video input and output as well as analog genlock input and loop through as seen in Figure 2-1. Please note that In Link B and Out Link B are not implemented in the standard configuration of the SGI Media Server.

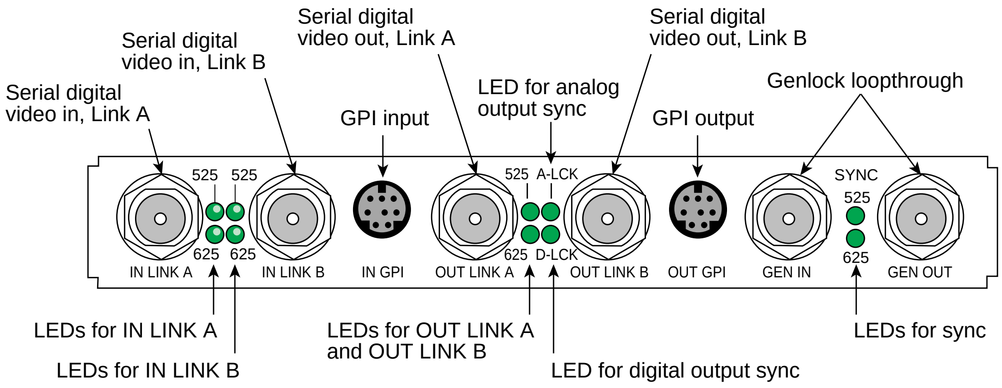

Serial digital video in, Link A
Serial digital video in, Link B
Serial digital video out, Link A
GPI input
LED for analog output sync
Serial digital video out, Link B
GPI output
Genlock loopthrough
525
525
625
625
IN LINK A
IN LINK B
IN GPI
525
A-LCK
625
D-LCK
OUT LINK A
OUT LINK B
OUT GPI
SYNC
525
625
GEN IN
GEN OUT
LEDs for IN LINK A
LEDs for IN LINK B
LEDs for OUT LINK A and OUT LINK B
LED for digital output sync
LEDs for sync

Figure 2-1 DIVO-DVC Panel

Table 2-3 summarizes DIVO-DVC board external connectors that interface with video equipment.

Table 2-3 Interface for Video Equipment

Connector	Format	Use
IN LINK A **IN LINK B**	10-bit CCIR 601-2, 75-ohm BNCs terminated, unbalanced	Serial digital video input from digital tape deck or other recording device. Conforms to SMPTE 259M for component video, SMPTE 272M for embedded audio, and SMPTE 266M for DVITC. Both inputs autophased.
OUT LINK A **OUT LINK B**	10-bit CCIR 601-2, 75-ohm BNCs	Serial digital video output to digital tape deck or other recording device. Conforms to SMPTE 259M for component video, SMPTE 272M for embedded audio, and SMPTE 266M for DVITC.
GEN IN	75-ohm BNC, loopthrough, unbalanced, unterminated	External analog sync source (precision time base or other source of house sync) or analog loopthrough.
GEN OUT	75-ohm BNC loopthrough, unbalanced, unterminated	External reference loop out; passive loopthrough for genlock input with buffered signal to workstation. **Note**: If you attach a cable to one GEN connector, you must attach either another cable to other equipment accepting analog sync or a 75-ohm BNC terminator to the other GEN connector.
IN GPI **OUT GPI**	8-pin mini-DIN	Not implemented in the SGI Media Server.

Table 2-4 summarizes the function of the LEDs on the panel.

Table 2-4 DIVO-DVC Panel LEDs

LED	Purpose
Leftmost LEDs between **IN LINK A** and **IN LINK B** (**525** and **625**)	Top LED lights when valid 525-line serial digital signal detected on **IN LINK A**. Bottom LED lights when valid 625-line serial digital signal detected on **IN LINK A**.
Rightmost LEDs between **IN LINK A** and **IN LINK B** (**525** and **625**)	Valid 525-line (top LED) or 625-line (bottom LED) serial digital signal detected on **IN LINK B**.
Leftmost LEDs between **OUT LINK A** and **OUT LINK B** (**525** and **625**)	Valid 525-line (top LED) or 625-line (bottom LED) serial digital signal detected on **OUT LINK A** and **OUT LINK B**; these outputs are locked together, regardless of whether **OUT LINK B** is used.
A-LCK	Output is locked to an analog source: standalone, genlock (to another sync source), or free run.
D-LCK	Output is locked to a digital source (**IN LINK A** or **IN LINK B**).
SYNC 525 and **625**	Valid 525-line sync source (top LED) or 625-line sync source (bottom LED) detected.

System Configuration

This chapter contains two sections:

- "Configuring the SGI Media Server" on page 25
- "Video Server Technology Controls" on page 38

Configuring the SGI Media Server

This section describes how to configure the SGI Media Server in the following sections:

- "Configuring the Device Interface" on page 26
- "Configuring System Defaults" on page 27
- "Configuring Audio Support" on page 27
- "Configuring the Server for Control by Remote Devices (control-in.conf File)" on page 29
- "Configuring the Server to Control Other Devices (control-out.conf File)" on page 34
- "Mapping Physical Ports to Logical Port Names" on page 36
- "Configuring for 625/50 Systems" on page 36
- "Tuning the Media Server (Setting the Maximum DMA Size)" on page 36
- "Configuration Files" on page 38

Note: To configure startup options, see "Starting and Stopping VST" on page 57 in Chapter 4.

Configuring the Device Interface

Device files contain variables called controls that influence the behavior of specific devices. Each device in your system can have only one device file.

Configuring Audio and Video Devices in Your System

The directory */usr/vtr/config/device-defaults* contains files that correspond to the media devices used by the media server; these files define machine-specific behavior. Each device or device type has one file; the directory *device-defaults* can contain the following files:

- *MEDIA*, for behavior common to all connected media devices
- *DIVO_DVC*, for DIVO-DVC boards
- *dq*, or *dq_0*, *dq_1*, *dq_2*, and so on. when the media server is using Diaquest software to control external VTRs

Setting Device Parameters

You use the VST controls to configure video compression modes, audio sampling parameters, clip formats, and many other elements of the system configuration. For example, for the first DIVO_DVC device, you would create in the */usr/vtr/configd/device-defaults* directory a file called *DIVO_DVC_0*. This file could contain, for example:

```
# Format
vtr.media.video.input.compression.type dvcpro
vtr.media.clip.format movie/dif

# Audio
vtr.media.audio.input.port RAD1.AESIn
vtr.media.audio.output.port RAD1.AESOut
```

This example device defaults file sets the default control values for any units created using the DIVO_DVC_0 port so that new clips are recorded with DVCPRO compression and the audio input and output uses the AES interfaces on the Digital Audio Option board.

Configuring System Defaults

You use VST controls in files in the */usr/vtr/config/system-defaults* directory to specify default system-wide behavior. For example, the file *main* might contain the following:

```
vtr.main.timing_standard 625
```

This line sets the default timing mode for the VST system to 625.

Refer to "Video Server Technology Controls" on page 38 or to the vst-controls(5) man page for detailed information about all system controls.

Configuring Audio Support

For the DIVO-DVC output, a unit can select the audio port for the following:

- audio output during playback
- audio input during recording

The controls that configure the audio port are

- vtr.media.audio.output.port (default DefaultOut)
- vtr.media.audio.input.port (default DefaultIn)

The default audio input and output ports are selected with the IRIX Audiopanel control panel. Audio configuration is managed by the decoder.

For more information about audio ports, see the vst-controls(5M) man page.

Identifying Audio Ports

The default input audio port is DefaultIn, meaning the audio is taken from the default input as set in the server's audio panel. Similarly, the default output audio port is DefaultOut.

To change the input or output audio port, set either vtr.media.audio.input.port or vtr.media.audio.output.port, respectively. Audio port names take the form [*subsystem.*]*interface*. You must specify subsystem if the server has more than one audio device installed.

For audio ports on the SGI Digital Audio Option PCI board:

- *subsystem* is RAD*n*
- *interface* is AESIn, AESOut, ADATIn, or ADATOut

Each Digital Audio Option board has a unique subsystem name, such as RAD1 or RAD2. To determine the name of a board in a particular slot, use *vtrhwinfo*. Here is a slightly abbreviated example:

```
# /usr/vtr/bin/vtrhwinfo
System:
  Platform: Origin 200 GIGAchannel (hostname=system1 serial=xxxx)
  Processors: 2x225MHz R10000 (IP27)
  Secondary Cache: 2 Mbytes
  Memory: 512 Mbytes
Video devices:
  DIVO_DVC_0 (module=1/GIGAchannel slot=12 serial=HTM962
part=030-1387-001E)
    DVCPRO_CODEC (serial=HTM769 part=030-1388-001B)
    DVCPRO_CODEC (serial=HTM802 part=030-1388-001B)
  DIVO_DVC_1 (module=1/GIGAchannel slot=13 serial=HTM970 .
.
.
.
Audio devices:
  RAD1 (module=1/GIGAchannel pci=2)
  RAD2 (module=1/O200 pci=5)
Audio input ports:
  DIVO_DVC_0.DigitalIn
  DIVO_DVC_1.DigitalIn
  DIVO_DVC_2.DigitalIn
  DIVO_DVC_3.DigitalIn
  RAD1.AESIn
  RAD1.ADATIn
  RAD2.AESIn
  RAD2.ADATIn
Audio output ports:
  DIVO_DVC_0.DigitalOut
  DIVO_DVC_1.DigitalOut
  DIVO_DVC_2.DigitalOut
  DIVO_DVC_3.DigitalOut
  RAD1.AESOut
  RAD1.ADATOut
  RAD2.AESOut
  RAD2.ADATOut
```

```
Serial ports:
  tty1 (module=1/O200 built-in)
  tty2 (module=1/O200 built-in)
  tty3 (module=1/GIGAchannel slot=11)
  ...
```

For example, to take audio input for the DIVO_1 port from the AES coaxial input on the Digital Audio Option board identified as RAD1, open the file *`/usr/vtr/config/device-defaults/DIVO_1`*, and add this control setting:

```
vtr.media.audio.input.port RAD1.AESIn
```

Disabling Audio

If no audio input is connected, you must disable audio before recording. Add the following control setting to the appropriate device-defaults file (for example, *`/usr/vtr/config/device-defaults/DIVO_DVC_2`*):

```
vtr.media.audio.input.port ““
```

Configuring the Server for Control by Remote Devices (control-in.conf File)

Use the *`/usr/vtr/config/control-in.conf`* file to configure the media server so that it can be controlled by remote devices. The media server supports devices that use either the Louth Video Disk Communications Protocol (VDCP) or the industry-standard Sony-compatible VTR RS-422 control protocol.

The *`/usr/vtr/config/control-in.conf`* file has three parts:

- control port (serial port) configuration: one line per device
- other control ports: one line per device
- signal port configuration: one line per device

The control port and signal configuration lines are described in separate subsections below.

Control Port Configuration Line

You use the configuration file *`/usr/vtr/config/control-in.conf`* to specify basic connection parameters between the controlling devices (Sony and Louth) and the media server. Connection parameters include such information as which serial port controls which video board and how.

For each serial port to be connected to a controlling device (such as an automation or edit controller), insert a line in */usr/vtr/config/control-in.conf* that uses the following format:

```
protocol   type  port  speed   parity     signalport    rate    latency
```

The following line shows a sample configuration:

```
sony    rs422    4    38400    1    1    29.97    0
```

This example specifies that a controlling device using the Sony-compatible VTR protocol is connected via an RS-422 serial line to serial port 4 on the server. The serial connection speed is 38400 bits / sec, using odd parity. The device is controlling the video port identified by the “signal 1” configuration line (also specified in *control-in.conf*) and is using a zero-frame command latency, so that commands are executed as quickly as possible. The preroll should always be 0, except for Louth controllers, for which the preroll is 3.

The following subsections describe each variable in this configuration line.

protocol

Use the value that corresponds to the protocol used by the device you are configuring to work with the media server. The valid values of *protocol_name* include:

- sony, p2: Sony-compatible VTR RS-422 control protocol
- louth: Louth Video Disk Communications Protocol
- mvcp: SGI Multiport Video Computer Protocol
- hsip: Horita Serial Interface Protocol
- littlered: Miranda Little Red linear timecode reader protocol

type

The valid values of *type* include:

- rs232: RS-232 control connection
- rs422: RS-422 control connection

- tcp,: socket-based TCP / IP connection

These values must correspond to the type of connection between the control video device and the media server. The type of cable you use to connect the video device and the server determines this value.

port

The value of *port* specifies the serial port number for RS-232 and RS-422 connections. For example, to specify that the device is connected to port 2 (sometimes referred to as ttyd2), specify just the numeral 2. For TCP connections, *port* specifies the TCP port number.

Note: Serial and TCP ports (communications ports) are unrelated to video ports (signal ports).

---

speed

The *speed* value specifies the connection speed of the serial connection between the server and the controlling device, usually 38400. On rare occasions, 9600 is used. For TCP connections, enter a hyphen (-) for this setting.

parity

The optional *parity* setting can have one of the following values:

- 0, indicating no parity
- 1, indicating odd parity
- 2, indicating even parity

Parity is usually odd (1). On rare occasions, even or no parity is used.

signalport

The *signalport* value specifies the signal (video) port as identified by a signal configuration line in the `control-in.conf` that is controlled by this remote device connection. For more information, see “Signal Configuration Line” on page 33.

rate

The *rate* value specifies the frame-rate and drop-frame mode for time codes exchanged over the control connection. The following are the valid values:

- 25: 625 / 50 video
- 29.97: 525 / 59.94 video (drop-frame)
- 30: 525 / 59.94 video (non-drop-frame)

latency

When the media server receives a command, the command can apply only to the next frame to enter the video port output queue. For example, if the video port output queue holds three frames, the first frame that can be affected by an incoming command is the third frame in the queue. If a stop command is received, the first two frames in the video port output queue are displayed; the third is not displayed because it is affected by the stop command.

You can modify the number of frames that display before a command begins by setting the *latency* value. Valid values are 0 or positive integers. A value of 0 starts the command's effect as soon as possible, but the command latency is always greater than 0.

The minimum command latency is the sum of the fixed latency value for the specific compression format and the value of the vtr.media.video.output.max_queued_frames control, which is usually 1 (see the description of this control in the man page, vstcontrols (5M)). To guarantee that all commands are executed in a frame-accurate manner, add 1 to this sum, as follows:

- fixed latency value for the compression format:

  DVCPRO: 5 frames
- plus 1 (usual value of the vtr.media.video.output.max_queued_frames control)
- plus 1 (to guarantee that all commands are executed in a frame-accurate manner)

The result is 7 frames for DVCPRO.

A small number for *latency* increases the risk that a frame may be missed (repeated) because of system load. A large number increases the latency in responding to transport commands such as play, stop, or pause.

Some commands in the Sony protocol imply a predefined frame delay; *latency* does not affect these commands.

For the MVCP TCP port, make sure that the *control-in.conf* file includes a control port configuration line:

```
mvcp tcp 5250
```

5250 is the standard MVCP control port number. The initial *control-in.conf* file installed with the VST software includes this line.

Signal Configuration Line

The signal configuration line specifies how to map signal port numbers to video port names. A signal configuration line is required for each video port for it to be controlled by an external device. For example, to specify that signal 1 corresponds to the video port known as DIVO_DVC_4, add the following line:

```
signal 1 DIVO_DVC_4
```

Example control-in File

Here is a sample *control-in.conf* file:

```
# This defines the normal VST MVCP TCP port

mvcp tcp 5250

# Other control ports

#sony   rs422 2 38400 1 1 29.97 3
louth   rs422 2 38400 1 1 29.97 3
louth   rs422 5 38400 1 2 29.97 3
#
louth   rs422 6 38400 1 3 29.97 3
louth   rs422 7 38400 1 4 29.97 3
#louth   rs422 8 38400 1 5 29.97 3
#
#
# Signal ports
#
#   Each signal (video) port is mapped from a port number to port name
#   with a line of this form:
#
```

```
#   signal <port number> <port name>
#
#     port number:      Disk port number (> 0)
#     port name:        VST port name (e.g., DIVO_DVC_1)

signal 1 DIVO_DVC_0
signal 2 DIVO_DVC_1
signal 3 DIVO_DVC_2
signal 4 DIVO_DVC_3
```

Configuring the Server to Control Other Devices (control-out.conf File)

Use the *`/usr/vtr/config/control-out.conf`* file to configure the server so that it can control other VTR-like devices.

Use this file also to specify basic connection parameters between the srver and the VTRs to be controlled. Connection parameters include such information as which serial port controls which VTR and how. For each VTR controlled by the server using the Sony protocol, add a line in *`/usr/vtr/config/control-out.conf`* that uses the following format:

```
protocol    connection    serial_port     rate    parity
```

The following line shows a sample configuration:

```
sony    rs422    4    38400    1
```

The line has the same meaning when p2 is substituted for sony.

The following sections describe each variable in this configuration line.

protocol

Use the value that corresponds to the device you are configuring to work with the server. The valid values of *protocol_name* are sony and p2, which are equivalent.

connection

The valid values of *connection* include:

- rs232, for a RS-232 control connection
- rs422, for a RS-422 control connection

These values must correspond with the type of connection between the video device and the server. The device and the server must agree on the serial connection; the cable between them must be appropriate for that connection.

serial_port

The *serial_port* value specifies the serial port as identified by a serial configuration line in the `control-in.conf` that is controlled by this remote device connection.

Note that The RS-422 pinout at the DB-9 end of the SGI adapter cable is nonstandard; see Appendix I.

speed

The *speed* value specifies the connection speed of the serial connection between the server and the controlling device, usually 38400. On rare occasions, 9600 is used. For TCP connections, enter a hyphen (-) for this setting. Sony devices must use 38400.

parity

The optional *parity* setting can have one of the following values:

- 0, indicating no parity
- 1, indicating odd parity
- 2, indicating even parity

The parity must be specified as odd for the currently supported protocols.

Mapping Physical Ports to Logical Port Names

You can use the `/usr/vtr/ports.conf` file to create logical names for the physical video port s on the media server. For each port, include a line in the configuration file with this syntax:

`port` *physicalname logicalname description*

For example:

```
port DIVO_DVC_0 ProgA “Program A”
port DIVO_DVC_1 ProgB “Program B”
port DIVO_DVC_2 Preview “Preview Output”
port DIVO_DVC_3 Archive “Archive I/O”
```

Configuring for 625/50 Systems

The media server supports both 525 / 59.94 and 625 / 50 operation simultaneously. A few operating characteristics default to 525 / 59.94, including the MVCP timing mode (refer to the MVCP *FRAT* command documentation in Appendix H, “Multiport Video Computer Protocol (MVCP) Command Summary,”) and the default video output timing.

To change the default timing, include the following control setting in the system defaults file */usr/vtr/config/system-defaults/main*:

```
vtr.main.timing_standard 625
```

Note: The system timing standard controls the default video output timing only if the control vtr.media.video.output.timing is not changed from its default value of “system”.

In */usr/vtr/config/device-defaults/DIVO_DVC*, make sure vtr.media.video.output.timing is set to its default value, system.

Tuning the Media Server (Setting the Maximum DMA Size)

The maximum DMA size limits the maximum amount of data that can be transferred to or from the filesystem in a single operation. The default maximum DMA size is 4 MB, which is too small for most installations. To increase the system's maximum DMA size, use the following procedure:

1. As superuser, start *systune* to change the DMA size:

   ```
   # systune -i
   ```

   The system responds:

   ```
   Updates will be made to running system and /unix.install

   systune->
   ```

2. Enter the new DMA size:

   ```
   systune-> maxdmasz bytes
   ```

   The following example shows 10 MB:

   ```
   systune-> maxdmasz 641
   ```

   The system responds:

   ```
        maxdmasz = 257 (0x101)
        Do you really want to change maxdmasz to 640? (y/n)
   ```

3. Enter **y** for yes. The system responds:

   ```
   In order for the change in parameter maxdmasz to become effective,
   reboot the system
   ```

4. Quit *systune*:

   ```
   systune-> quit
   ```

5. Reboot the server:

   ```
   # reboot
   ```

Configuration Files

Table 3-1 summarizes the files that the server uses so that it works with all the hardware and software in the system.

Table 3-1 Configuration Files

Configuration File	Purpose
/usr/vtr/config/control-in.conf	Configures the server to be controlled by remote devices. Use this file to configure VCP to work with controllers compatible with Sony and Louth protocols, and Miranda or Horita time-code reader devices.
/usr/vtr/config/control-out.conf	Configures the server to control VTRs or VTR-like devices.
/usr/vtr/config/control-out.conf	Configures the server to control other decks. Use this file to configure the server to control VTRs using Diaquest software.
/usr/vtr/config/device-defaults	This directory contains files that specify default control settings for audio, video, and disk devices. The names and contents of the files are described in “Video Server Technology Controls” on page 38.
/usr/vtr/config/system-defaults	This directory contains files that specify default control settings for various non-device subsystems (for example, system-wide controls or Sony protocol control ports).
/usr/vtr/config/studiocentral.conf	Specifies StudioCentral repositories that the server can access as archive systems.
/usr/vtr/config/ports.conf	Specifies logical video port names to substitute for the ports’ physical names.
/usr/vtr/config/vtrd.conf	The server server daemon configuration.
/usr/vtr/config/system-defaults/clipmirror	Stores the name of the primary server with which redundant server synchronizes, and other parameters. See Appendix F.

Video Server Technology Controls

Video Server Technology (VST) controls are configuration, control, and status variables implemented by VST system and device interface modules. These controls provide an easily extensible mechanism for configuring and controlling device-independent and device-specific features.

Note: This appendix contains tables summarizing the controls. For complete information on all controls, see the controls man page, vstcontrols(5M).

The controls fall into the following categories:

- "System Controls" on page 39
- "Device Controls" on page 44

Note: In the Default column in the tables in this appendix, the entry

(none)

signifies that the control has no default setting. (Parentheses also enclose other explanatory comments in the Default column.) An entry of

none

signifies that the word "none" is the default.

System Controls

System controls are configuration variables not associated with a particular device or unit. These values can be set at startup with a system defaults file. After startup, they can also be queried and set by using the MVCP commands *SGET* and *SSET*, respectively.

A system defaults file resides in */usr/vtr/config/system-defaults* and is named after the subsystem. For example, */usr/vtr/config/system-defaults/main* holds the settings for the main system defaults.

The rest of this section explains the controls, in these subsections:

- "System Controls for Clip Mirroring" on page 40
- "System Controls for Filesystems" on page 40

- "System Controls for the Main System" on page 41
- "System Controls for the Time Subsystem" on page 42
- "System Controls for Sony Protocol Devices (VTR Emulation)" on page 43

System Controls for Clip Mirroring

Table 3-2 summarizes clip mirror controls. The defaults file is */usr/vtr/config/system-defaults/clipmirror*.

Table 3-2 System Controls for Clip Mirroring

Control	Default	Use
vtr.clipmirror.primary_server. hostname	"" (null string; mirroring disabled	Specifies name of server to mirror and starts local mirroring of the specified server's clip cache. Unsetting this control halts mirroring. See "Setting Up Primary and Redundant Servers" on page 219 in Appendix F.
vtr.clipmirror.local_server. hostname	"" (null string; mirroring disabled	Specifies name of redundant server. See "Setting Up Primary and Redundant Servers" on page 219 in Appendix F.
vtr.clipmirror.max_threads	20	Specifies the number of concurrent ftp threads allowed. Enter any integer from 0 to 100, based on the available network bandwidth between the two servers.
vtr.clipmirror.reconnect_interval	30	Specifies how often (in seconds) the clip mirror server tries to reconnect to the primary server after the connection is lost. Enter any integer greater than 0.

System Controls for Filesystems

Table 3-3 summarizes filesystem controls. The defaults file is */usr/vtr/config/system-defaults/fs*.

Table 3-3 System Controls for Filesystems

Control	Default	Use
vtr.storage.fs.grio.enabled	true	Enables or disables use of Guaranteed Rate I/O (GRIO).
vtr.storage.fs.grio.quantum_increment	50000	Specifies the increment in microseconds used to round down GRIO time quantums for reservations.

Table 3-3 (continued) System Controls for Filesystems

Control	Default	Use
vtr.storage.fs.grio.reap_interval	10000000000	Specifies the interval (in nanoseconds) for deallocating unused GRIO bandwidth reservations.
vtr.storage.fs.grio.reserve_mode {standard | maximum | average}	standard	Specifies disk bandwidth reserved for accessing a clip.
vtr.storage.fs.io_size	0 (not specified)	Specifies minimum number of bytes read from or written to a clip file.
vtr.storage.fs.latency_warning_threshold	400	Specifies threshold (in milliseconds) for a warning message if the threshold is exceeded by a single I/O operation during playback or recording.
vtr.storage.fs.major_alignment	0 (not specified)	Specifies alignment boundary used to optimize access to clip file.
vtr.storage.fs.max_cue_reservations	1	Specifies maximum bandwidth reservations per real-time filesystem for cueing clips for playback.
vtr.storage.fs.max_io_vectors	0 (use system maximum)	Specifies maximum I/O vectors used in a single I/O operation.
vtr.storage.fs.minor_alignment	0 (not specified)	Specifies alignment boundary used to match the access requirements of the clip file.

System Controls for the Main System

Table 3-4 summarizes main system controls. The defaults file is *`/usr/vtr/config/system-defaults/main`*.

Table 3-4 System Controls for the Main System

Control	Default	Use
.vtr.main.heap.dump	0	Internal control; do not change.
.vtr.main.heap.trace_heap_pid	0	Internal control; do not change.
vtr.main.heap.trace_pid	0	Internal control; do not change.
.vtr.main.list_threads	0	Internal control; do not change.
.vtr.main.ulock_spin_count	0	Internal control; do not change.

Table 3-4 (continued) System Controls for the Main System

Control	Default	Use
.vtr.main.ulock_yield_count	0	Internal control; do not change.
vtr.main.log_level.console	0	Specifies maximum log level of messages sent to stdout.
vtr.main.log_level.file	0	Specifies maximum log level of messages sent to server log file (usually, */usr/vtr/adm/logs/vtrlog*).
vtr.main.log_level.syslog	0	Specifies maximum log level of messages sent to system log (usually */var/adm/SYSLOG*).
vtr.main.thread.cpu_limit	0	Internal control; do not change.
vtr.main.thread.cpu_limit_interval	10	Internal control; do not change.
vtr.main.timing_standard	525	Specifies default timing standard for system (525 or 625).
vtr.main.unit_heap_size	8388608	Sets size (in bytes) of the heap allocated to each unit.

System Controls for the Time Subsystem

Table 3-5 summarizes main system controls. The defaults file is */usr/vtr/config/system-defaults/time*.

Table 3-5 System Controls for the Time Subsystem

Control	Default	Use
vtr.time.slave_system_time	true (for channel 1, unless another channel specifically enabled)	Specifies whether system time (maintained by IRIX) is slaved to timecode inputs for this time channel.
vtr.time.offset	-27000000 (Miranda Little Red) -6600000 (Horita PR-232)	Sets offset (in nanoseconds) between actual timebase and decoded input timecode.

System Controls for Sony Protocol Devices (VTR Emulation)

Table 3-6 summarizes device controls for devices that use the Sony protocol.

Note: Settings in the defaults file `/usr/vtr/config/system-defaults/sony` apply to all Sony-protocol control ports. Settings that apply to individual control ports are in the file `/usr/vtr/config/system-defaults/sony_`*n*, where *n* is the serial port number.

Table 3-6 System Controls for Sony Protocol Devices

Control	Default	Use
vtr.control.clip.name	(none)	Specifies clip to load on a Sony-protocol-controlled port.
vtr.control.default_id.name	(none)	Specifies default ID (clip) to be loaded on a port controlled by the Sony protocol.
vtr.control.description	(none)	Specifies a descriptive name for the control port.
vtr.control.device_type_id	0xd803	Specifies device type identifier response to a DEVICE-TYPE protocol request.
vtr.control.edit.delay	5	Specifies delay after an EDIT_ON or EDIT_OFF command is received before specified operation commences.
vtr.control.ee.delay	5	Specifies delay after an EE_ON or EE_OFF command is received before the specified operation commences.
vtr.control.ee.mode	player	Specifies whether the video output is delayed so that seamless playback/EE switching occurs (recorder mode), or whether video output matches time code reported by VST (player mode).
vtr.control.ee.record_select	true	Specifies whether output is switched to EE mode during recording.
vtr.control.output.idle_mode	hold	Specifies mode of output port when no playback is occurring.
vtr.control.preread	false	Unsupported.
vtr.control.sony.input_discard	true	Specifies whether input port is always active and discarding frames when no recording is occurring.
vtr.control.sony.log_level	3	Specifies log level of control processor debugging messages.

Table 3-6 (continued) System Controls for Sony Protocol Devices

Control	Default	Use
vtr.control.sony.timecode_log_level	0	Specifies log level of control processor time-code debugging messages.
vtr.control.superimpose.enabled	false	Do not change.
vtr.control.timecode.mode	drop-frame	Specifies whether control port reports "drop-frame" or "non-drop-frame" time code for 525/59.94 video.
vtr.edit.postroll	3:00	Specifies the postroll for an AUTOEDIT operation.
vtr.edit.preroll	5:00	Specifies the preroll for an AUTOEDIT operation.

Device Controls

Device controls are configuration, control, and status variables that 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 *GET* and *SET* commands.

A *device-defaults* file resides in */usr/vtr/config/device-defaults*. These 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:

1. Settings for all devices: ALL
2. Settings for node type: MEDIA or STORAGE
3. Settings for port type: VIDEO, NETWORK, DECK, or DISK
4. Settings for device class: *deviceclass*
5. Settings for device: *devicename*

For example, when the unit device interface for the DIVO_DVC_1 video device is initialized, settings are loaded from the device defaults files as follows:

1. ALL
2. MEDIA

3. VIDEO
4. DIVO_DVC
5. DIVO_DVC_1

This section discusses device controls in the following subsections:

- “Device Controls for DIVO-DVC Ports” on page 45
- “Device Controls for Deck Control (Diaquest) Ports” on page 52
- “Device Controls for Filesystem Access” on page 53

Device Controls for DIVO-DVC Ports

Table 3-7 summarizes device controls for supported video devices. The following is the path of the defaults file:

> *`/usr/vtr/config/device-defaults/DIVO_DVC`* for a DIVO-DVC board; *`/usr/vtr/config/device-defaults/DIVO-DVC_n`* is the port-specific version of the file

Table 3-7 Device Controls for DIVO-DVC Ports

Control	Default	Use
vtr.edit.postroll	3:00	Specifies postroll duration for an edit performed by the unit. Not the same as the postroll set for a machine control port (such as for the Sony or Louth protocol).
vtr.edit.preroll	5:00	Specifies preroll duration for an edit performed by the unit. Not the same as the preroll set for a machine control port (such as for the Sony or Louth protocol).
vtr.media.audio.input.channel_map {<map> | *}	*	Specifies mapping from device input audio channels to clip audio channels.
vtr.media.audio.input.channels	2	Specifies number of audio input channels to be read from audio device.
vtr.media.audio.input.channels.clip	-1	Specifies number of audio channels to put in a new clip.
vtr.media.audio.input.edge_detect	0	Enables an internal test mode.
vtr.media.audio.input.port	DefaultIn	Specifies audio input port.
vtr.media.audio.input.rate	48000	Specifies audio sampling rate in samples per second.

Table 3-7 (continued) Device Controls for DIVO-DVC Ports

Control	Default	Use
vtr.media.audio.input.sample.format	twos-complement	Specifies format of audio samples.
vtr.media.audio.input.sample.width	24	Specifies the number of bits of precision in each audio sample.
vtr.media.audio.input.skew	0	Specifies a skew value (in nanoseconds) for adjusting synchronization between input audio and video streams.
vtr.media.audio.input.sync_source	<blank>	Specifies sync source for clocking input audio.
vtr.media.audio.output.channel_map {<map> | *}	*	Specifies mapping from clip audio channels to device output audio channels.
vtr.media.audio.output.channels	2	Specifies number of audio output channels to be written to audio device.
vtr.media.audio.output.fade. duration.in	10000000	Specifies time (in nanoseconds) for fading in audio at a clip transition point.
vtr.media.audio.output.fade.duration. out	10000000	Specifies nanoseconds for fading out audio at a clip transition point.
vtr.media.audio.output.mute	false	Specifies whether normal audio samples are sent to attached audio port or whether zero samples (digital silence) are sent.
vtr.media.audio.output.port	DefaultOut	Specifies audio output port.
vtr.media.audio.output.rate	48000	Specifies audio sampling rate in samples per second.
vtr.media.audio.output.sample.format	twos-complement	Specifies format of audio samples.
vtr.media.audio.output.sample.width	24	Specifies bits of precision in each audio sample.
vtr.media.audio.output.skew	0	Specifies a skew value (in nanoseconds) for adjusting synchronization between output audio and video streams.
vtr.media.audio.output.sync_source	Video	Specifies sync source for clocking output audio.
vtr.media.audio.output.tone.frequency	1000	Specifies frequency (in Hz) of tone output when unit output mode is set to “image”.
vtr.media.audio.output.tone.level	-20	Specifies level (in dB) of tone output when unit output mode is set to “image”.
vtr.media.clip.format	default	Specifies format of new clips.

Table 3-7 (continued) Device Controls for DIVO-DVC Ports

Control	Default	Use
vtr.media.clip.limit.enabled	true	Specifies whether starting or ending limits are enforced when clip is played back.
vtr.media.clip.limit.end	*	Sets upper limit (end) for unit.
vtr.media.clip.limit.start	*	Sets lower limit (start) for unit.
vtr.media.clip.location.preset	* (none)	Sets location to which clip should be cued if playback or recording is started without the unit’s being manually cued.
vtr.media.clip.segmented.format	(none)	Sets format for creating segmented clips.
vtr.media.clip.segmented.insert_mode	overwrite	Specifies whether a new segment recorded into a segmented clip overwrites the existing frames in the clip, or whether it is inserted into clip at cued location.
vtr.media.clip.start.mode	preset	Specifies mode for determining time code of first frame recorded into an empty clip if no time code is specified when unit is cued for recording.
vtr.media.clip.start.preset	01:00:00:00	Specifies time code of first frame recorded into an empty clip if no time code is specified when unit is cued for recording and vtr.media.clip.start.mode is set to “preset”.
vtr.media.ee.offset_frames	0	Sets offset (in video frames) between input video/audio and output video/audio when EE mode is active.
vtr.media.ee_follows_record	false	Not supported.
vtr.media.input. abort_on_dropped_frame	false	Specifies whether a record operation is automatically aborted if an input frame is dropped.
vtr.media.input.buffer_depth	3	Specifies memory for buffering incoming video data before it is written to storage.
vtr.media.input.frame_log_level	4	Sets logging level for input frame debugging messages.
vtr.media.input.idle_mode	off	Sets idle mode behavior for input port.
vtr.media.input.mode	store	Sets recording behavior for input port.
vtr.media.input.preemptible	true	Specifies whether unit can be preempted while idle if idle mode is set to a value other than “off”.

Table 3-7 (continued) Device Controls for DIVO-DVC Ports

Control	Default	Use
vtr.media.input.timecode.preset	0	Sets initial value of free-running counter used by VITC and user bits free-running mode.
vtr.media.input.trigger.mode.in	none	Sets type of trigger used to begin recording.
vtr.media.input.trigger.mode.out	none	Sets type of trigger used to end recording.
vtr.media.input.trigger.vitc.in	(none)	Sets VITC of first frame recorded when input trigger-in mode is VITC.
vtr.media.input.trigger.vitc.out	(none)	Sets VITC of frame following last frame recorded when input trigger-out mode is VITC.
vtr.media.input.userbits.mode	source	Specifies data stored in the time code user bits for each recorded frame.
vtr.media.input.userbits.preset	0	Sets initial value of the free-running and running user bits counters.
vtr.media.output.buffer_depth	3	Specifies amount of memory for buffering outgoing video data after it is read from storage system.
vtr.media.output.cue_buffer_depth	1	Specifies amount of memory for buffering outgoing video data for a clip being cued to play after it is read from storage system.
vtr.media.output.cued_mode	output	Sets unit’s output mode when it is cued.
vtr.media.output.idle_mode	hold	Sets unit’s output mode when it is not playing.
vtr.media.output.mode	clip	Sets unit’s output mode when it is playing.
vtr.media.output.pause_at_limits	false	Specifies whether unit should pause (instead of stop) when start or end limit is reached.
vtr.media.output.preemptible	true	Specifies whether unit can be preempted while idle if idle mode is a value other than “off”.
vtr.media.output.speed.fast_forward	50000	Sets speed of playback when a fast forward (FF) command is issued.
vtr.media.output.speed.rewind	-50000	Sets speed of playback when a rewind (REW) command is issued.
vtr.media.output.timecode.preset	0	Sets initial value of free-running counter used by VITC and user bits free-running mode.

Table 3-7 (continued) Device Controls for DIVO-DVC Ports

Control	Default	Use
vtr.media.output.userbits.mode	source	Specifies data sent in the time code user bits of each frame.
vtr.media.output.userbits.preset	0	Sets initial value of the free-running and running user bits counters.
vtr.media.port.enabled	true	Specifies whether this media port is enabled for use by VST applications.
vtr.media.port.shared_access	true	Specifies whether access to this media port is shared between VST and other applications.
vtr.media.port.use_fillpoints	auto	Internal control; do not change.
vtr.media.video. beep_on_dropped_frame	no	Specifies whether a bell character (\007) is written to log when frame is dropped.
vtr.media.video.input.bit_rate. feedback_gain	0.01	Specifies magnitude of feedback applied in adjusting compression quality when recording a clip using a lossy compression method with a target bit rate.
vtr.media.video.input.bit_rate.target	25000000	Specifies target bit rate for compression stage when a clip is recorded using a lossy compression method.
vtr.media.video.input.colorspace	ccir601	Specifies color-space encoding for recording input video.
vtr.media.video.input.compare. command	(none)	Enables an internal diagnostics mode; do not change.
vtr.media.video.input.compare.host	(none)	Enables an internal diagnostics mode; do not change.
vtr.media.video.input.compression. dithering	off	Not currently implemented.
vtr.media.video.input.compression. precision	8	Specifies bits of precision used by compression algorithm to store video component data.
vtr.media.video.input.compression. quality	0.94	Specifies value of quality parameter supplied to a lossy compression algorithm.
vtr.media.video.input.compression. sampling	422	Specifies type of video component sampling used by compression algorithm.
vtr.media.video.input.compression. type	default	Specifies type of compression for video frames being recorded.

Table 3-7 (continued) Device Controls for DIVO-DVC Ports

Control	Default	Use
vtr.media.video.input.field_dominance	f1	Specifies input field dominance.
vtr.media.video.input.format	normal	Specifies format of input video data.
vtr.media.video.input.frame_mode	false	For compression modes that support it, specifies whether input video fields are stored as interleaved frames.
vtr.media.video.input.hard_reset	false	When this control is set to true, the associated hardware input port is reset.
vtr.media.video.input. interface_precision	10	Specifies precision of DIVO-DVC Serial Digital Interface (SDI) inputs.
vtr.media.video.input.multiplex. transmission_rate	4	Specifies transmission rate expected for input from a multiplexed video stream (such as SDTI).
vtr.media.video.input.packing	R242_8	Specifies video component packing.
vtr.media.video.input.restart_attempts	4	Specifies number of times VST tries to restart a failed input video transfer (recording) before raising an error on the unit.
vtr.media.video.input.signal_loss. timeout	30000000000 (30 seconds)	Specifies how long (in nanoseconds) VST waits for a video input signal to be reacquired before signalling error condition on active unit.
vtr.media.video.input.source	normal	Specifies source for input video.
vtr.media.video.input.timing	auto	Specifies timing for input video.
vtr.media.video.input.vitc.mode	source	Specifies mode for storing VITC for frames as they are recorded into a clip.
vtr.media.video.input.vitc_line_offset	14	Specifies video line used to get VITC from input video signal.
vtr.media.video.output.boot_image. blank.pixel	0x28005c00	Specifies RGBA pixel value for DIVO-DVC output when video port is reset and vtr.media.video.output.boot_image.type is set to “blank”.
vtr.media.video.output.boot_image. info.text	$port	Specifies text displayed in center of DIVO-DVC output when video port is reset and vtr.media.video.output.boot_image.type is set to “info”.
vtr.media.video.output.boot_image. type	info	Specifies initial DIVO-DVC output when video port is reset.

Table 3-7 (continued) Device Controls for DIVO-DVC Ports

Control	Default	Use
vtr.media.video.output.fast_shuttle_repeat_count	2	Specifies number of times each displayed frame is repeated when unit is playing forward or backward at a speed greater than normal.
vtr.media.video.output.field_dominance	f1	Specifies output video field dominance.
vtr.media.video.output.format	normal	Specifies format of output video data.
vtr.media.video.output.hard_reset	false	When this control is set to true, the associated hardware output port is reset.
vtr.media.video.output.image.name	black	Specifies image displayed when the output mode is “image” and vtr.media.output.image.type is “user”.
vtr.media.video.output.image.type	bars	Specifies image displayed when unit output mode is “image”.
vtr.media.video.output.interface_precision	10	Specifies precision of DIVO-DVC Serial Digital Interface (SDI) outputs.
vtr.media.video.output.max_queued_frames	1	Specifies maximum video frames queued for video output.
vtr.media.video.output.multiplex.transmission_rate	4	Specifies transmission rate for output to a multiplexed video stream (such as SDTI).
vtr.media.video.output.overlay.mode	fast	Specifies whether text overlay should be done with as little CPU overhead as possible or so that output text is always clean.
vtr.media.video.output.overlay.offset.horizontal	64	Specifies horizontal pixel offset of left edge of text overlay.
vtr.media.video.output.overlay.offset.vertical	-64	Specifies horizontal pixel offset of upper edge of text overlay.
vtr.media.video.output.overlay.text	(none)	Specifies text of overlay drawn into each outgoing video field.
vtr.media.video.output.repeat_both_fields	no	Specifies whether both fields of video frame are displayed when frame is repeated.
vtr.media.video.output.restart_attempts	4	Specifies the number of times VST tries to restart a failed output video transfer (playback) before raising an error on the unit.

Table 3-7 (continued) Device Controls for DIVO-DVC Ports

Control	Default	Use
vtr.media.video.output.skip_on_drop	true	When one or more frames are dropped (and the previous frame is repeated) because of a delay in providing the next frame to the video port, specifies whether the same number of frames are automatically skipped in the clip so that clip playback duration does not change.
vtr.media.video.output.sync.offset.vertical	0	Specifies vertical offset (in lines) applied to output video relative to incoming sync reference.
vtr.media.video.output.sync_source	external	Specifies source of synchronization for video output.
vtr.media.video.output.timing	system	Specifies default timing standard for video output.
vtr.media.video.output.vitc.line_offset	14	Specifies video line where VITC is placed in output signal.
vtr.media.video.output.vitc.mode	clip	Specifies data sent for VITC of each frame.

Device Controls for Deck Control (Diaquest) Ports

Table 3-8 summarizes deck and editing device (Diaquest) controls. The defaults file is */usr/vtr/config/device-defaults/dq*.

Table 3-8 Device Controls for Deck Control Ports

Control	Default	Read Only	Use Summary
vtr.deck.error	(none)	Yes	Returns last error detected from deck control interface.
vtr.deck.status	(none)	Yes	Returns current deck transport status.
vtr.edit.coincidence.preroll	10	No	Specifies how far in advance of edit in point that VST attempts to sync. Under some circumstances, the actual coincidence point is a few frames later than specified, so this control should be set to at least 10 frames.
vtr.edit.preroll	5:00	No	Specifies deck preroll for edit operations.
vtr.edit.postroll	3:00	No	Specifies deck postroll for edit operations.
vtr.edit.status	{none)	Yes	Returns current status of edit operation that is in progress.

Table 3-8 (continued) Device Controls for Deck Control Ports

Control	Default	Read Only	Use Summary
vtr.media.video.frame_rate	{none)	Yes	Returns video frame rate reported by deck.
vtr.media.video.sync_port	{none)	No	Specifies port number from which the deck control interface should derive video sync timing.
vtr.media.output.mode	pb	No	Specifies current output mode of deck (pb for normal playback, ee for E-to-E).

Device Controls for Filesystem Access

Table 3-9 summarizes device controls for storage devices. The defaults file is *`/usr/vtr/config/device-defaults/fs`*.

Table 3-9 Device Controls for Filesystem Access

Control	Default	Use
vtr.storage.clear_after_play	false	Specifies that frames are automatically cleared from clip after they are played.
vtr.storage.continue_after_error	true	Specifies that the unit continues playing after an I/O error.
vtr.storage.io_log_level	4	Specifies log level for I/O logging messages.
vtr.storage.max_open_files	8	Specifies maximum number of clip files that the unit can have open concurrently.

PART TWO

Operations II

This part of the book describes operational tasks involved in running your media server.

Chapter 4, “Running the SGI Media Server,” describes starting, monitoring, and stopping the Video Server Technology (VST) software.

Chapter 5, “Using the SGI Media Server Control GUI,” describes the graphical user interface (GUI) that you use to control and status system activities like playing and recording clips.

Chapter 6, “Adding and Removing Clips,” describes how you add and remove clips from the media server.

Chapter 7, “Using Clip Manager,” describes how to use the Clip Manager to manipulate clips and their attributes.

Chapter 8, “Virtual Clips,” describes virtual clips and how you manipulate them on the media server.

Chapter 9, “Configuring and Using External Devices,” describes how to handle the external devices connected to the media server.

Chapter 4

Running the SGI Media Server

This chapter consists of the following:

- "Starting and Stopping VST" on page 57
- "VST Man Pages" on page 60
- "Logging Events" on page 61
- "Monitoring the System" on page 64

Starting and Stopping VST

Video Server Technology (VST) is normally started automatically when the server boots..

This section explains how to start VST in the following sections:

- "Configuring VST to Start Automatically" on page 57
- "Starting VST Manually" on page 58
- "Stopping VST" on page 58
- "Setting Startup Options" on page 59
- "Checking VST Status" on page 59

Note: For information on booting from a backup plex, see "Configuring for Booting From a Backup Plex" in Chapter 10.

Configuring VST to Start Automatically

By default, when VST is installed, it does not enable itself to be automatically started when the system boots. As superuser, you can configure VST to start automatically at boot time using *chkconfig* with the **vtr** option. The following enables automatic startup:

```
# chkconfig vtr on
```

The following disables automatic startup:

```
# chkconfig vtr off
```

Starting VST Manually

You must be root to start VST. Start VST with

```
# /usr/vtr/bin/vtrstart
vtrstart: Starting video server
```

Note: The message is displayed only if the system has verbose logging enabled (*chkconfig verbose on*).

If VST is already running, *vtrstart* displays an error:

```
vtrstart: Video server is already running (use -f to force restart)
```

As indicated, to stop the VST instance currently running and start a new VST instance, add the -f option:

```
# /usr/vtr/bin/vtrstart -f
vtrstart: Stopping video server
vtrstart: Starting video server
```

Stopping VST

Stop VST with

```
# /usr/vtr/bin/vtrstop
vtrstart: Stopping video server
```

Setting Startup Options

When the VST daemon *vtrd* starts, it reads the configuration file */usr/vtr/config/vtrd.conf*. The daemon *vtrd* is the VST server daemon that manages the VST server processes.

The default installation operates normally without any change to the startup options. However, you can change the startup options for *vvtr*, the VST server process, by editing */usr/vtr/config/vtrd.conf* and adding them immediately after the program path, */usr/vtr/bin/vvtr*.

Table 4-1 describes the startup options for *vvtr*.

Table 4-1 VST Server Process (*vvtr*) Startup Options

Option	Description
-l *logopts*	Sets the log options, which control the format of the messages written to the log. If this option is omitted, each message contains the severity code, a timestamp, the process ID, and the text of the message. The following may be specified for this option: -l l, do not include the severity code in each log message -l t, do not include the time stamp in each log message -l p, do not include the process ID in each log message See “VST Log Message Structure” on page 61 for more information about log message format.
-p	Specifies that VST should not run at high priority (p). If this option is omitted, VST runs at high real-time system priority to ensure no interruptions in video / audio input or output. Include this option only at the direction of SGI Media Server support personnel.

Checking VST Status

You can use *vtrstat* to check whether VST is running on the system. If VST is not running, *vtrstat* displays the following message:

```
# vtrstat
Video server on <host> is stopped
```

If VST is running and responding to MVCP connections, *vtrstat* displays

```
Video server on <host> is running.
```

For additional VST status information displayed by `vtrstat`, see “Monitoring the System” on page 64.

VST Man Pages

Table D-2 summarizes VST man pages:

Table D-2 Man Pages

Man Page	Describes
vst(1)	Video Server Toolkit
vtrstart(1)	Startup
vtrstop(1)	Shutdown
vtrstat(1)	Status
vtrclip(1)	Utility for adding and removing clips to or from clip cache
mcpanel(1)	Media control panel
mcclips(1)	Clip manager
mcstat(1)	Status display
mccompstats(1)	Compression monitor
vstcontrols(5)	VST controls
mvcp(5)	Multiport Video Computer Protocol
vvtr(1)	VST server
vtrd(1)	VST daemon
vtrvfutil(1)	VST vframe clip utility
vtrftpd(1)	VST FTP daemon

You can list the VST man pages after installing them by entering the following command:

```
% versions long vst_eoe | grep man
```

Logging Events

The VST logging feature provides a mechanism for storing a text record of events that occur during operation of the system. The default logging configuration generally records only events that are generated by various error conditions, but you can enable additional logging to help track and trouble-shoot VST behavior.

This section explains the following:

- “VST Log Message Structure” on page 61
- “VST Man Pages” on page 60
- “Configuring Logging” on page 63
- “Managing Log Rollover” on page 63

VST Log Message Structure

Each VST log message has the following format:

c dd-hh:mm:ss.mmmmmm pppp <log message>

where

- *c* is the severity code of the message
- *dd* is the day of the month
- *hh:mm:ss.mmmmmm* is a time stamp that indicates when the message was written
- *pppp* is the process ID of the process that wrote the message to the log
- *log message* is the actual text message

Table 4-3 shows the VST log severity levels and codes, which are listed in decreasing order of severity.

Table 4-3 Log Levels

Severity Level	Severity Code	Description
Emergency	P	Panic condition.
Alert	A	A condition that should be corrected immediately, such as a corrupted system file.
Critical	C	A critical condition that has system-wide impact, such as a hard device error; immediate action is required.
Error	E	A problem that needs correcting, but does not require immediate action.
Warning	W	Possible problem, but could be a transient problem that corrects itself.
Notice	N	Condition that might require attention, but isn't an error condition.
Info	I	Informational message.
Debug	*n*	Information message that normally is of use only to engineers for debugging; can be Debug1, Debug2, or Debug3, with Debug3 producing the most amount of debugging information.

The following is an example of a message that has a severity code of 2 (Debug2 severity level). The message was written on day 14 of the month at the time that is shown in the message, and the ID of the process that wrote the message is 8254:

```
2 14-22:23:50.316766  8254 mvcp/ninety9 <-- 100 VTR Ready
```

The following example shows a Notice-level message.

```
N 29-09:26:11.490919   3064 U1 Unit ERROR (err=22): Clip timecode type
not know
```

The following example shows an Info-level message.

```
I 29-09:26:12.758644   3065 (littlered_1) Timecode input acquired
tc=09:27:35.02 (offset=* -27000000)
```

The following example shows a Warning-level message.

```
W 28-14:06:56.807983  35339 U16(DIVO_DVC_2) Video input missing
```

Configuring Logging

Logging is configured by setting the value of the following controls in the system defaults file */usr/vtr/config/system-defaults/main*:

- vtr.main.log_level.file
- vtr.main.log_level.syslog

For example, to enable increased logging by including the first level of debug events in the events that are logged to the server log, */usr/vtr/adm/logs/vtrlog*, include the following control setting in the main system defaults file:

```
vtr.main.log_level.file 1
```

The value of the control is relative to the Info log event severity. Positive values enable more verbose logging (1 = Debug1, 2 = Debug2, ...); negative values disable all but the more severe events (-2 = Warning, -3 = Error, ...).

Managing Log Rollover

Log rollover is the practice of saving the current log file and restarting logging into an empty log file. Rollover avoids the problems of running out of disk space and of having a list of log messages too long to handle easily.

VST manages the vtrlog server log file. By default, the log file is saved and a new log file begun every night at 2:00 a.m. if the log file is at least 10 MB. The default number of log files retained on the system is ten.

The command *rotatelogs* is used to manage the log files and is executed by the system's cron job handler at the right time. Table 4-4 shows the list of available options.

Table 4-4 rotatelogs Command Options

Option	Specifies	Default Value
-b	Log filename	vtrlog
-d	Log file directory	/var/adm/vtr/logs
-h	Lists these options	
-l	Daemon to notify	vtrd

Table 4-4 rotatelogs Command Options

Option	Specifies	Default Value
-m	Maximum number of backups retained on the server	10
-s	Minimum log file size to trigger rollover	10 MB
-D	Debug level	Off

To change any defaults (such as the time of the rollover or the log size), the crontab entry of rotatelogs in the system must be edited (as root, using *crontab -e*).

For example, to retain the last 20 log files on a machine, the rotatelogs entry in crontab must look like:

```
1       2       *       *       *       /usr/vtr/bin/rotatelogs -m 20
```

The debug option is off by default. It can be turned on using the **-D** option. There is only one level of debugging information (on).

Monitoring the System

Once Video Server Technology (VST) is installed and running, you can monitor its operation using the tools mentioned in this chapter. Two tools, *vtrstat* and *mcstat*, work directly with VST.

This chapter describes how to monitor VST in the following sections:

- "vtrstat" on page 64
- "mcstat" on page 66

You can also use standard SGI IRIX tools to monitor various system resources. See Appendix D, "IRIX Monitoring Tools," for details.

vtrstat

vtrstat is a command-line tool that tells you:

- Whether or not VST is running

- Which units are open, if the **-units** option is used
- Which media ports are available, if the **-ports** option is used

Example 4-1 shows an example output of *vtrstat*.

Example 4-1 vtrstat Output

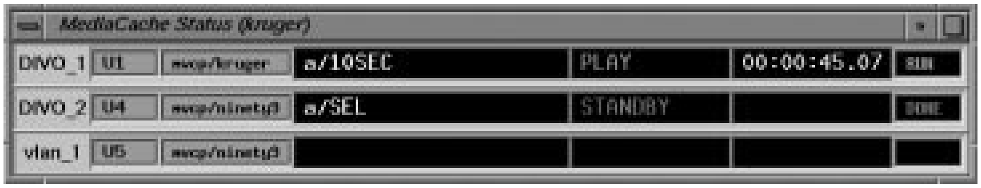
```
vsta 7# /usr/vtr/bin/vtrstat -units
Video server on vsta is running.

Unit  Owner        Port    Clip       Function Location
----------------------------------------------------------
U1    louth        DIVO_0  d/REC12    STOP     01:04:30.05
U2    louth        DIVO_0  *          STOP     *
U3    louth        DIVO_1  d/rPOP     CUE      00:00:00.00
U4    louth        DIVO_1  d/rSEL     STOP     00:00:00.00
U5    louth        DIVO_1  d/REC01    STOP     01:04:30.06
U6    louth        DIVO_1  *          STOP     *
U7    louth        DIVO_0  d/rSEL     CUE      00:00:00.00
U8    louth        DIVO_0  d/rSTOK    STOP     00:03:30.01
U9    louth        DIVO_7  d/rJOR     CUE      00:00:00.00
U10   louth        DIVO_7  d/rSEL     STOP     00:00:00.00
U11   louth        DIVO_3  d/rPOP     CUE      00:00:00.00
U12   louth        DIVO_3  d/rSTOK    PLAY     00:03:02.29
U13   louth        DIVO_4  d/rSEL     CUE      00:00:00.00
U14   louth        DIVO_4  d/rSTOK    STOP     00:00:00.00
U15   louth        DIVO_5  d/rSEL     CUE      00:00:00.00
U16   louth        DIVO_5  d/rSTOK    STOP     00:03:30.01

vsta 8# /usr/vtr/bin/vtrstat -ports
Video server on vsta is running.

#  Port    Type    Description
----------------------------------------------------------
1  DIVO_0  Video   SGI XT-DIVO Digital Video Option
2  DIVO_1  Video   SGI XT-DIVO Digital Video Option
3  DIVO_3  Video   SGI XT-DIVO Digital Video Option
4  DIVO_4  Video   SGI XT-DIVO Digital Video Option
5  DIVO_5  Video   SGI XT-DIVO Digital Video Option
6  DIVO_6  Video   SGI XT-DIVO Digital Video Option
7  DIVO_7  Video   SGI XT-DIVO Digital Video Option
8  dq_0    Deck    Diaquest Deck Control
```

A description of these functions can be found in Chapter 11, “Completing Common Tasks Using MVCP Commands.””

mcstat

mcstat graphically displays the activity of all units that are currently open on a the server. It is recommended that you install *vst_eoe.sw.base* and *vst_eoe.sw.tools* on the SGI workstation you are using to monitor the VST server and run *mcstat* on that workstation instead of the server.

Figure 4-1 shows an example of *mcstat*’s output when VST is running. When VST is not running, *mcstat* exits.



```
DIVO_1  U1  mvcp/kruger   a/10SEC  PLAY     00:00:45.07  RUN
DIVO_2  U4  mvcp/ninety3  a/SEL    STANDBY               DONE
vlan_1  U5  mvcp/ninety3
```

Figure 4-1 mcstat Output

mcstat displays a list of all the media ports supported by the server. For each port, it displays all the units that are opened on that port; each unit is on a separate line.

The unit display includes the name of the unit, the unit owner information, the name of the clip currently loaded on the unit, the last function executed by the unit (or the one currently executing), the current clip time code, and the status of the function.

Using the SGI Media Server Control GUI

SGI Media Server control graphical user interface (GUI) consists of the following:

- Media and Deck Control Panel (mcpanel), which enables *clip*s to be played and recorded.
- Unit Status Monitor (mcstat), which displays the status of media server ports.
- Clip Manager (mcclips), which is used to manage clips, including getting them from, and writing them to a StudioCentral 2.0 archive system.

The following topics are discussed in this chapter:

- "Monitoring the Status of the Media Server (mcstat)" on page 92Playing and Recording Clips with mcpanel

This section describes how to use the Media Server Control Panel (mcpanel)and how to control a video deck. The following topics are discussed:

- "Starting the Media Server Control Panel" on page 68
- "Determining Available Ports" on page 71
- "About the Media Server Control Panel" on page 74
- "Playing or Recording an Existing Clip" on page 81
- "Creating a New Clip" on page 83
- "Changing Cue Points and Edit Points" on page 84
- "Controlling a Video Deck" on page 85

See Chapter 6, "Adding and Removing Clips,"for information about how to copy digital media data from a file into the media server *clip cache*.

Starting the Media Server Control Panel

To establish a control connection to VST and start the Media Control Panel, enter the following, either from the workstation on the media server or from a workstation on which the VST tools software[1] has been installed:

```
% /usr/vtr/bin/mcpanel  -h hostname  -p videoPort|unit
```

For more complete control of mcpanel, use it with the following set of flags:

```
% /usr/vtr/bin/mcpanel  [-h hostname]  [-p videoPort|unit]  [-D deckCtlPort]
[-c clipname]  [-r]  [-C "inpoint outpoint"]  [-P]  [-v loglevel]
```

Table 5-1 describes each of the options available when starting the Media Server Control Panel.

Table 5-1 Media Server Control Panel Options

Option	**Description**
-c	Identifies the name of a clip to be loaded when the Media Server Control Panel starts. If this option is not specified, no clip is initially loaded.
-D	Specifies the Diaquest deck control port (deckCtlport) to be used for communication with an external video storage device, such as a digital videotape recorder (VTR). The deck is controlled by the Deck Control Panel. If this option is not specified, deck control is not available.
-h	Identifies the host on which VST runs. This enables the Media Server Control Panel to be run from a remote workstation. If this option is not specified, the local host is assumed.

[1] To run the Media Control Panel from a remote workstation, the vst_eoe.sw32.tools subsystem must be installed on a workstation that has IRIX 6.5. For more information, see Appendix C, “Installing Video Server Technology.”.

Table 5-1 Media Server Control Panel Options

Option	Description
-p	Identifies the media server video *port* or *unit* to which the control connection is made. *videoPort* is the media server video port to which the control connection is made. If you specify a port, VST creates a new logical unit that is used by this control connection. *unit* is the VST logical unit to which the connection is made. If you specify a unit, it must have already been created by another control connection. The control connection being made shares the unit with the control connection that created the unit. (Units are named with a capital "U" followed by a number, for example, U4.) If this option is not specified, the first video port on the media server is used.
-r	Specifies that if an mcpanel already exists for the video port, the existing mcpanel should be raised on the desktop instead of creating a new one.
-C	Specifies that the loaded clip should be cued with the specified in- and out-points. If "*" is specified for either *inpoint* or *outpoint*, the default edit in-point or out-point is used.
-P	Specifies that the clip whose name is *clipname* should start playing when the Media Server Control Panel starts.
-v	Sets the logging verbosity level to *loglevel*. The default is 0, meaning all log messages up to and including Info priority are written to STDOUT. (The mcpanel program writes its log messages to STDOUT.) The values for *loglevel* are defined in Table 5-2.

Table 5-2 shows the log severity levels and codes, which are listed in decreasing order of severity

Table 5-2 Log Severity Levels

Priority	Log Level	Description
Emergency	-6	Panic condition.
Alert	-5	Condition that should be corrected immediately, such as a corrupted system file.
Critical	-4	Critical condition that has system-wide impact, such as a hard device error; immediate action required.

Table 5-2 Log Severity Levels

Priority	Log Level	Description
Error	-3	Problem that needs correcting but does not require immediate action.
Warning	-2	Possible problem but could be a transient problem that corrects itself.
Notice	-1	Condition that might require attention, but is not an error condition.
Info	0	Informational message.
Debug*n*	*n*	Informational message that normally is of use to engineers for debugging; may be Debug1, Debug2, or Debug3, with Debug3 producing the most debugging information.

Priority levels Info and Notice result from user-caused actions. Priority levels Warning through Emergency generally result from system problems.

The Media Server Control Panel, shown in Figure 5-1, is displayed in its own window. This control panel represents a unit, or logical *VTR*, that is used to play and record *clip*s. The buttons in the control Panel are similar in function to those of a standard VTR. For example, there are buttons to load a clip, play it, and pause.

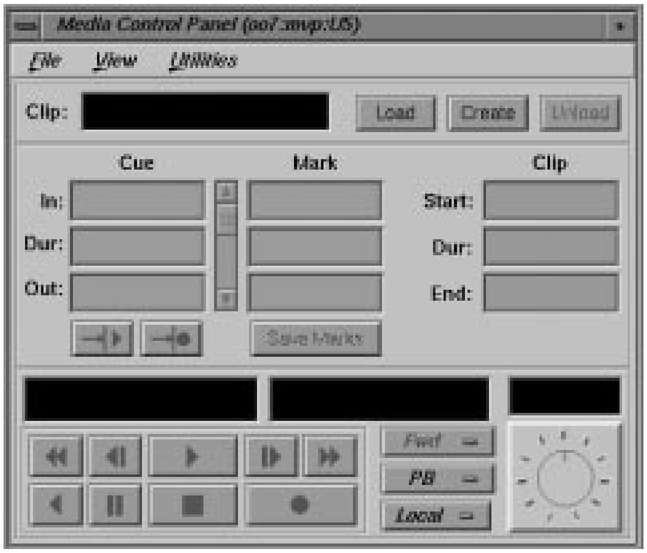


Figure 5-1 Media Server Control Panel

See "About the Media Server Control Panel" on page 74 for a detailed description of the Media Server Control Panel.

Determining Available Ports

To determine which video and deck control ports are available on a media server, use the `/usr/vtr/bin/vtrstat` command as follows:

```
% /usr/vtr/bin/vtrstat -ports
#  Port    Type    Description
----------------------------------------------------------
0  mvp     Video   SGI O2Video (Multiport Video Processor)
```

See the vtrstat(1) man page for more information.

Using Telnet to Determine Available Ports

To determine which video and deck control ports are available on the media server, establish a telnet connection to the server and then use the MVCP *PLS* (List Ports) command. (See "Establishing an Interactive MVCP Connection" on page 235 for details.)

The following example shows the *PLS* command output. This output identifies "vlan_1" as the deck control port (DECK) and "DIVO_*n*," where *n* is 0-7, as the video ports (VID):

```
PLS
201 OK
DIVO_DVC_0 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_0
DIVO_DVC_1 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_1
DIVO_DVC_2 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_2
DIVO_DVC_3 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_3
DIVO_DVC_4 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_4
DIVO_DVC_5 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_5
DIVO_DVC_6 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_6
DIVO_DVC_7 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_7
```

See "PLS" on page 245 for more information.

Determining Units in Use

To determine which units are in use on the media server, use the `/usr/vtr/bin/vtrstat` command as follows:

```
% /usr/vtr/bin/vtrstat -units
Unit  Owner        Port    Clip       Function Location
-----------------------------------------------------------
U3    mvcp/oo7     mvp     *          IDLE     *
```

See the vtrstat(1) man page for more information.

Using Telnet to Determine Units in Use

To determine which units are in use on the media server, establish a telnet connection to the server and then use the MVCP *ULS* (List Units) command. The following example shows the information returned by the *ULS* command:

```
ULS
201 OK
U1 mvcp/mediaserver mvp BOTH * DONE IDLE * 0 *
U2 mvcp/mediaserver mvp BOTH * DONE IDLE * 0 *
```

This example indicates that there are two units (U1 and U2) on the server, both using the mvcp port on the host named mediaserver. The units were opened for input and output (BOTH) and they are currently idle.

See "ULS" on page 247 for more information.

Starting the Media Server Control Panel

The following are examples of starting the Media Server Control Panel:

- When the Media Server Control Panel is started by entering the following command, a control connection is made to the DIVO_DVC_1 video *port* on the "media_server" host using a newly added unit. Messages with a severity level of Info and above are written on STDOUT:

  ```
  % /usr/vtr/bin/mcpanel  -h media_server  -p DIVO_DVC_1
  ```

- When the Media Server Control Panel is started by entering the following command, a control connection is made to the U9 unit, which must already exist. Messages with a severity level of Debug2 and above are written on STDOUT:

  ```
  % /usr/vtr/bin/mcpanel  -v 2  -h media_server  -p U9
  ```

About the Media Server Control Panel

Figure 5-2 shows the appearance of the control panel after a *clip* has been loaded. The header of the control panel identifies the host, control *port*, and *unit* to which the control panel is connected.

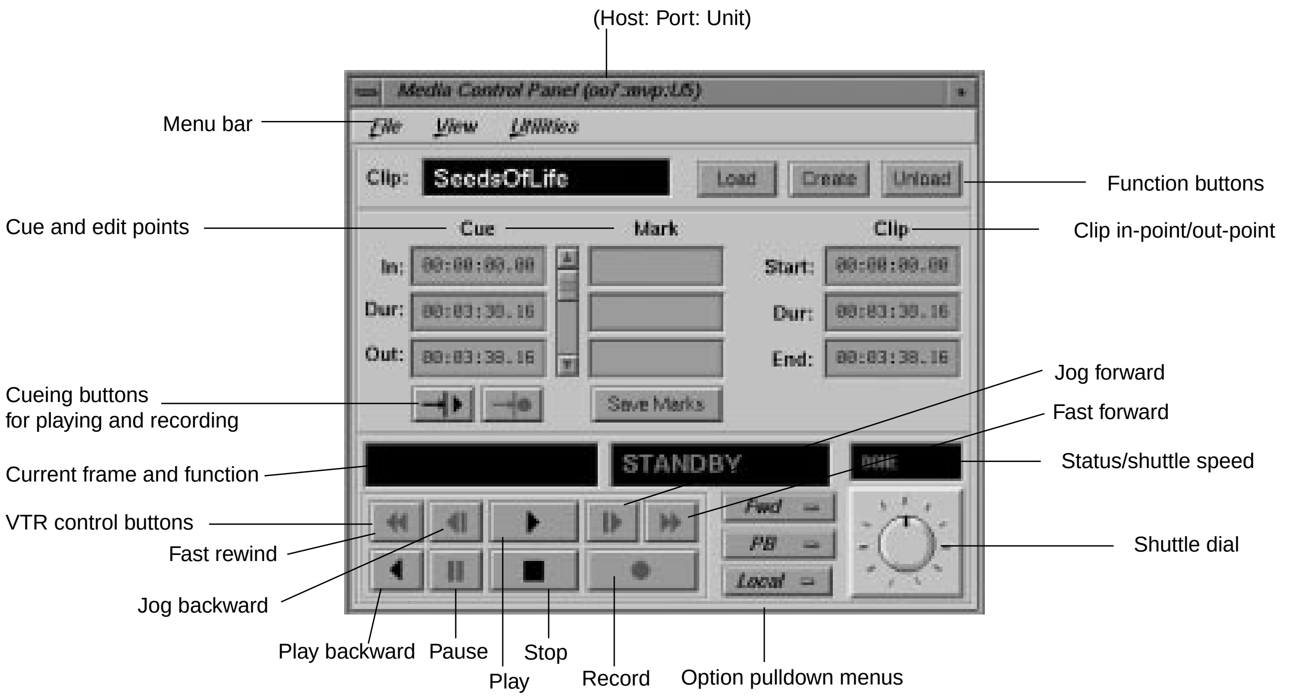


Figure 5-2 Media Server Control Panel With a Clip Loaded

The menu bar gives you access to the following:

- The File pulldown menu, which lets you load or unload an existing *clip*, create a new clip, or close the control panel. Most of the functions available in this pulldown menu are available through buttons in the control panel.
- The View pulldown menu, which lets you access the Deck Control window. The Deck Control window, shown in Figure 5-5, is used to control a video deck attached to the media server. See "About the Deck Control Window" on page 87 for more information.
- The Utilities pulldown menu, which lets you access the following:
  - IRIX audio panel
  - IRIX video panel
  - Unit Status Monitor (mcstat), described in "Monitoring the Status of the Media Server (mcstat)" on page 92
  - Clip Manager (mcclips), described in Chapter 7, "Using Clip Manager."

The following describes each of the displays and buttons in the control panel:

- The Clip field contains the name of the *clip*, if one is loaded. The display is blank if a clip is not loaded.
- The *Load*, *Create*, and *Unload* function buttons let you load an existing clip, create a new one and record into it, and unload a clip, respectively.
- The *cue points* are used to move around within a clip and to control the portion of the clip that is played. An in-point (In), the duration (Dur), and an out-point (Out) are specified using the following format:

  hh:mm:ss:ff

  where *hh* is the hours, *mm* is the minutes, *ss* is the seconds, and *ff* is the frame number. In drop-frame mode, the final colon is replaced by a period:

  hh:mm:ss.ff

  For example, if a clip with a cue in-point of 00:00:30.00 is cued for playing, it is cued at thirty seconds.

  See "Changing Cue Points and Edit Points" on page 84 for information about how to change the cue points.

- The *edit points* (under “Mark” in the Media Server Control Panel) are persistent values stored with a clip. They are used to initialize the *cue points* when the clip is loaded. An in-point (In), the duration (Dur), and an out-point (Out) are specified using the following format:

  hh:mm:ss:ff

  where *hh* is the hours, *mm* is the minutes, *ss* is the seconds, and *ff* is the frame number. In drop-frame mode, the final colon is replaced by a period:

  hh:mm:ss.ff

  Note: Edit points may also be referred to as “edit marks.”

  If a clip has edit points associated with it, those values are used to initialize the clip’s cue points when the clip is loaded. The cue points are often different from the start and end points of the clip.

  See “Changing Cue Points and Edit Points” on page 84 for information about how to change the edit points.

- To the right of the cue and edit points are the start, the duration (Dur), and end of the clip. Each is specified in the following format:

  hh:mm:ss:ff

  where *hh* is the hours, *mm* is the minutes, *ss* is the seconds, and *ff* is the frame number. In drop-frame mode, the final colon is replaced by a period, as in:

  hh:mm:ss.ff

  If a clip does not have edit points associated with it, these start, duration, and end values are used to initialize the cue points when the clip is loaded.

  Note: You cannot change the values of the start, end, or duration of the clip itself. However, you can select any of the values and copy it to the cue points or edit points.

- The cue buttons cue the clip for playout (—> | >) or recording (—> | ●). The location at which the clip is cued depends on how the clip is cued and the play direction, as discussed in the descriptions of the *VTR* control buttons and the Media Server Control Panel’s Option pulldown menus. (The play direction is determined by the setting of the topmost Option pulldown menu.)

Note: A clip must be cued before it can be played or recorded. This can be accomplished explicitly by clicking a cue button or implicitly by clicking the play or record button without first clicking a cue button. If you click a cue button and then click play or record, the clip starts playing or recording immediately. If you click play or record without first clicking a cue button, the Media Server Control Panel first cues the clip and then starts the requested function. In the latter case, there is a brief delay before the requested play or record function begins.

- The current frame display is initialized to the cue in-point or cue out-point each time the clip is cued. The actual *cue point* depends on how the clip is cued and the clip's play direction, as discussed in the descriptions of the *VTR* control buttons and the play direction. (The play direction is determined by the setting of the topmost Option pulldown menu.) As the clip is played or recorded, the display changes to indicate the current frame number.
- The function display shows the current function. When a clip is first loaded, the word "STANDBY" is displayed.
- When a clip is being played or recorded, the status / shuttle speed display shows the speed. If the clip is not being played or recorded, the status is displayed. The status may be one of the following:
  - WAIT, when the function is waiting to execute or waiting for another *unit* to finish (for example, a Louth *automation controller* that is playing a clip). The word BUSY blinks on and off.
  - RUN, when the function is in progress.
  - DONE, when the function completed without an error. The word DONE and the function display are grayed-out.
  - ERROR, when an error has occurred. The word ERROR blinks on and off, and appears in red.

- The *VTR* control buttons control the playout and recording of clips. The following describes these buttons, which correspond to standard VTR buttons:
  - Fast-reverse (<<) plays the clip in reverse at a fast speed.
  - Jog backward (<|) jogs the clip backward by one frame. Each time you click the jog backward button, the clip jogs back one frame.

    If the clip is playing when you click this button, the clip jogs backward one frame and then pauses. You have to click either the play button or the pause button to resume play.
  - Forward play (>) plays the clip in the forward direction. If the clip is not cued, it is cued before it begins playout.
  - Jog forward (|>) jogs the clip forward by one frame. Each time you click the jog forward button, the clip advances one frame.

    If the clip is playing when you click this button, the clip jogs forward one frame and then pauses. You have to click either the play button or the pause button to resume play.
  - Fast-forward (>>) plays the clip forward at a fast speed.
  - Reverse play (<) plays the clip in reverse. If the clip is not cued, it is cued before it begins playout.
  - Pause (| |) temporarily stops the clip from playing or recording. You have to click the play, record, or pause button to resume.
  - Stop (■) stops the clip and de-cues it. After the clip has been stopped, you must re-cue it before playing it again.
  - Record (●) begins recording. If the clip is not cued, it is cued for recording before it begins.

- The top Option pulldown menu lets you specify the play direction, which determines the direction in which the clip is played and whether it plays once or plays until it is stopped. The following options are available through this menu:
  - *Fwd*, for forward play (default)
  - *F Lp*, for forward loop play
  - *Bwd*, for backward play
  - *B Lp*, for backward loop play
  - *F/Bwd*, for alternating forward and backward play
  - *F/B Lp*, for alternating forward and backward loop play
  - *B/Fwd*, for alternating backward and forward play
  - *B/F Lp*, for alternating backward and forward loop play
  - FCue, for forward play without cue (in and out) points set
  - BCue, for backward play without cue (in and out) points set

  If the direction is one of the forward directions (*Fwd*, *F Lp*, *F/Bwd*, or *F/B Lp*), the clip is cued at its in-point. If the direction is one of the backward directions (*Bwd*, *B Lp*, *B/Fwd*, or *B/F Lp)*, the clip is cued at its out-point.

  The forward direction means that the clip plays from its in-point to its out-point. The backward direction means that the clip plays from its out-point to its in-point. For alternating directions, the clip plays once in each direction. In loop mode, the clip continues to play in the given direction until you click the stop button. If you do not choose loop mode, the clip plays once in the indicated direction and then stops.

  Note: When you change the option in this menu, the change takes effect the next time you cue the clip. For example, assume that the *Fwd* option is in effect when you start playing a clip. If you choose the *F Lp* option after the playing starts, the play stops when the out-point is reached. The next time you play the clip, the clip plays in forward loop mode.

  To play forward or backward without using cue points, you first use FCue and BCue, respectively, to cue the clips. When clips limits are disabled using the controls, *vtr.media.clip.limit.start* and *vtr.media.clip.limit.end*, you can play forward or backward without limitation; past the beginning or end of a clip, however, the clip will play black.

- The middle Option pulldown menu lets you choose the following options:
  - *PB*, the clip’s audio and video are output when a clip is playing; nothing is output at other times (default).
  - *PB/EE*, the clip’s audio and video are output when a clip is playing; the input signal is output at other times.

    Note: The EE (end-to-end) option emulates a video deck feature and works only when you have an active input source.

  - *PB/Im*, the clip’s audio and video are output when a clip is playing; SMPTE 75% colors bars and 1 kHz tone are output at other times.
  - *PB/B*, the clip’s audio and video are output when a clip is playing; a black screen is output at other times.
  - *EE*, the output always displays the input signal instead of the signal from the media server, even when a clip is playing. (When a clip is playing, the output displays the input signal.)
  - *Image*, the output always displays SMPTE 75% colors bars and plays a 1 kHz tone.
  - *Black*, the output always displays a black screen.
  - *Hold*, the output always displays the last image.
- The Local/Rem Option pulldown menu lets you put the *unit* in remote mode. The following options are available through this menu:
  - *Local*, which puts the unit in local mode and enables the *VTR* control buttons (local).
  - *Rem*, which puts the unit in remote mode and disables the VTR control buttons. This mode prevents you from accidentally operating a Media Server Control Panel while the unit is being controlled remotely, for example, by an *automation controller.*
- The shuttle dial lets you control the speed of a clip that is playing. To use the shuttle dial, begin playing the clip and then use the mouse to point to the black notch on the dial. Press the left mouse button and keep it pressed while you turn the dial clockwise to increase the speed or counterclockwise to decrease it.

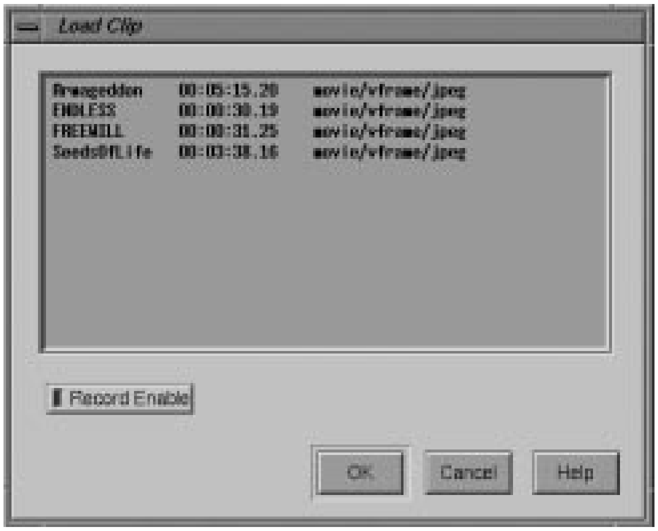
  Note: The straight up position of the dial is zero, or pause.

In general, buttons that are enabled and can be used appear darker than those that are disabled. For example, when you load an existing clip, the cue for playout button (—> | >) is darker than the cue for recording button (—> | ●). This indicates that the cue for playout button is enabled and the cue for recording button is not.

Playing or Recording an Existing Clip

To play or record an existing *clip*, follow these steps:

1. If the clip is not loaded, load the clip:
   - Click the *Load* button in the Media Server Control Panel. A window that lists the clips in the media server cache appears, as shown in Figure 5-3.



```
Armageddon     00:05:15.20    movie/vframe/jpeg
ENDLESS        00:00:30.19    movie/vframe/jpeg
FREEWILL       00:00:31.25    movie/vframe/jpeg
SeedsOfLife    00:03:38.16    movie/vframe/jpeg
```

Figure 5-3 Loading a Clip Into the Logical VTR

The Load Clip window lists the name, duration, and format of each clip that is stored under `/usr/vtr/clips`, which is the VST *clip cache*. For example, the SeedsOfLife clip has a duration of 00:03:38.16 and its format is movie/vframe/jpeg. (If the duration contains an asterisk [*], it means that no material has been recorded in the clip.)

Note: If a clip is stored in a directory within `/usr/vtr/clips`, that directory name precedes the clip name. For example, if the clip name appears as ADS/COMM1, the clip is stored in `/usr/vtr/clips/ADS/COMM1`.

- Select the clip that you want to load. (You select the clip by pointing to it with the mouse cursor and pressing the left mouse button.)
- If you want to record over this clip, click *Record Enable*. (When recording is enabled, the yellow LED is lit in the *Record Enable* button.) Click *Record Enable* a second time if you want to disable this option.
- Click the *OK* button. The window closes and the selected clip appears in the Media Server Control Panel. The cue for playout button (—>|>) is enabled. If the clip can be recorded over, the cue for recording button (—>|●) is also enabled.

2. Change the *cue point*s, as needed. (See “Changing Cue Points and Edit Points” on page 84 for information.)
3. Click the cue for playout button (-->|>) or the cue for recording button (—>|●).

Note: If you cue the clip first, the clip starts playing or recording as soon as you click the play or record button. If you do not cue the clip beforehand, the control panel automatically cues the clip and then starts playing or recording it. In the latter case, there is a delay before the requested function begins.

---

4. Click the play button (>) to start the playout or the record button (●) to start recording.
5. To stop playing or recording, click the stop button (■).

Note: If you want to replay or re-record the clip after it stops, the clip must be re-cued. If you click the play or record button without first clicking the cue button, the control panel automatically cues the clip before it starts playing or recording.

Creating a New Clip

To create a new *clip* and record its content, follow these steps:

1. Click the *Create* button in the Media Server Control Panel. The Create Clip window, shown in Figure 5-4, appears.

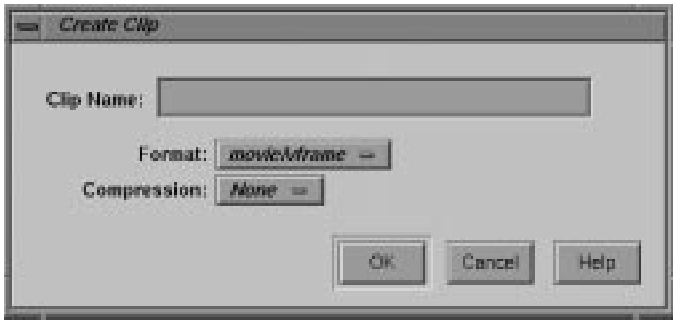


Figure 5-4 Create Clip Window

2. Enter the name of the clip in the Clip Name field.
3. Choose the compression type of the clip from the *Compression* Option pulldown menu. Choose DVCPRO compression.
4. Choose the format of the movie from the *Format* Option pulldown menu.

Note: You can create clips through the Media Server Control GUI using this option:

```
movie/dif/dvcpro
```

5. Click the *OK* button. The window closes and the clip appears in the control panel. Both the cue for recording button (—> | ●) and the cue for playout button (--> | >) are enabled.
6. Click the cue for recording button (—> | ●).
7. To start recording, click the record button (●).
8. To stop recording, click the stop button (■).

For more information about controlling the video deck, see "Controlling a Video Deck" on page 85.

Changing Cue Points and Edit Points

You can change the *cue point*s and *edit point*s of a clip in the Media Server Control Panel in three ways:

- Select and type a new value. For example, if you want to change a value from 00:00:00.01 to 00:00:00.05, select the “1” and type “5.”
- Click inside a field and use the scroll bar. The control panel automatically adjusts the other entries, accordingly, when you use the scroll bar. For example, to increase the cue in-point, click any place within the cue in-point field and then click the scroll bar to move it down. As the cue in-point increases, the duration decreases by the same amount.
- Select values with the left mouse button and then copy them using the middle mouse button.

If you press the Enter key after changing one of the values, the control panel verifies the new value. If the new value is valid, the control panel adjusts the other values to correspond to the new one, if necessary. If the new value is invalid, the control panel displays an error message in a dialog box.

If you do not press the Enter key after changing one of the values, the control panel does not check the validity of the new value. When you perform a function that uses the new values (for example, cueing the clip), the control panel then performs validity checking and displays an error message if a value is invalid.

Note: Changes that you make to the cue points take effect the next time the clip is cued. They have no affect on a clip that is already cued or is playing.

Any changes you make to the edit points or cue points are temporary. The changes cease to exist after the clip is unloaded unless you do one of the following:

- To make changes to the edit points permanent (that is, stored persistently with the clip), click the *Save Marks* button after you make the changes.
- To make changes to the cue points permanent, copy them to the edit points and then click the *Save Marks* button. The next time the clip is loaded, the newly saved edit points are copied to the cue points.

Note: Only the last set of edit points are saved. That is, when you save the edit points, they replace the edit points that are currently stored with the clip.

Controlling a Video Deck

This section describes how to use the Deck Control window to control a video deck that is attached to the media server. (For information on attaching a video deck to the media server, see Chapter 9, “Configuring and Using External Devices.”)

Note: To use the Deck Control window, you must specify a deck control port in addition to a video port when starting the Media Server Control Panel. (See “Starting the Media Server Control Panel” on page 68 for details.)

The following topics are discussed in this section:

- “Accessing the Deck Control Window” on page 86
- “About the Deck Control Window” on page 87
- “Recording From the Deck to a Clip” on page 90
- “Recording From a Clip to the Deck” on page 91

Accessing the Deck Control Window

To access the Deck Control window, choose View > Deck Control Panel from the menu bar of the Media Server Control Panel. The Deck Control window, shown in Figure 5-5, appears in a separate window.

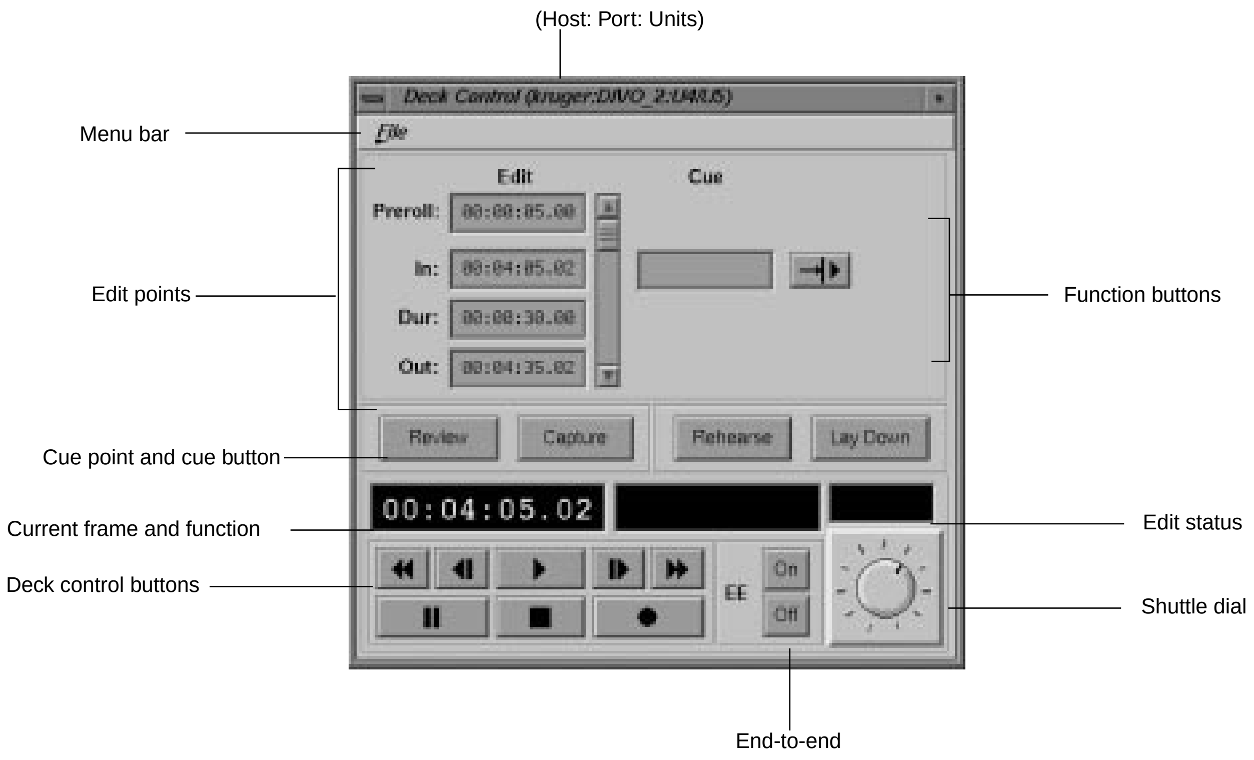


Figure 5-5 Deck Control Window

About the Deck Control Window

The File pulldown menu in the menu bar of the Deck Control window gives you access to the following options:

- Close, to close the Deck Control window
- Exit, to close both the Deck Control window and the control panel from which it was launched.

The following describes each of the displays and buttons in the Deck Control window:

- The *edit point*s let you set the amount of time to preroll, the in-point, the duration, and the out-point. Each of these can be specified in the following format:

  hh:mm:ss:ff

  where *hh* is the hours, *mm* is the minutes, *ss* is the seconds, and *ff* is the frame number. In drop-frame mode, the final colon is replaced by a period:

  hh:mm:ss.ff

  See "Changing Cue Points and Edit Points" on page 84 for information about how to change the edit points.

- The *cue point* is the location at which you want to position the deck. The cue point can be specified in the following format:

  hh:mm:ss:ff

  where *hh* is the hours, *mm* is the minutes, *ss* is the seconds, and *ff* is the frame number. In drop-frame mode, the final colon is replaced by a period:

  hh:mm:ss.ff

- The cue button (—> | >) tells the media server to search the deck to the cue point and park it there.

- You can use the following deck control function buttons:
  - *Review*, to review the specified edit. After the edit is finished, you can use this option to examine the finished edit.
  - *Capture*, to perform a frame-accurate recording from the selected portion of a loaded and cued source tape to the *clip* currently loaded. For more information about recording, see “Recording From the Deck to a Clip” on page 90.
  - *Rehearse*, to practice the specified edit. The decks go through all the mechanics of a real except that the edit is not recorded.
  - *Lay Down*, to perform a frame-accurate recording from the selected portion of the clip on the media server to the destination VTR. For more information, see “Recording From a Clip to the Deck” on page 91.
- The current frame display is initialized to the cue in-point each time the clip is cued. As the clip is played or recorded, the display changes to indicate the current frame number.
- The function display shows the current function.
- The edit status display shows the status when performing an automated edit (that is, a capture, review, lay-down, or rehearse). The status may be one of the following:
  - CUE, when searching for the *cue point*
  - SYNC, when doing a preroll
  - LOCK, when the deck transport is locked
  - EDIT, when the edit is in progress
  - DONE, when the edit is complete

- The deck control buttons, which control playout and recording, correspond to standard *VTR* buttons. The following describes these buttons:
  - Fast-reverse (<<) moves the deck in reverse at a fast speed.
  - Jog backward (<|) jogs the deck backward by one frame. Each time you click the jog backward button, the deck jogs back one frame.

    If the deck is playing when you click this button, the deck jogs backward one frame and then pauses. You have to click either the play button or the pause button to resume play.
  - Forward play (>) moves the deck in the forward direction. If the deck is not cued, it is cued before it begins playout.
  - Jog forward (|>) jogs the deck forward by one frame. Each time you click the jog forward button, the deck advances one frame.

    If the deck is playing when you click this button, the deck jogs forward one frame and then pauses. You have to click either the play button or the pause button to resume play.
  - Fast-forward (>>) plays the deck forward at a fast speed.
  - Pause (| |) temporarily stops the deck from playing or recording. You have to click the play, record, or pause button to resume.
  - Stop (■) stops the deck and de-cues it. After the deck has been stopped, you must re-cue it to play it again.
  - Record (●) begins recording.
- The EE *On* and *Off* buttons turn the deck's end-to-end mode on and off, respectively. If you click the *On* button, the output displays the input signal instead of the signal from the deck. If you click the *Off* button, the output displays the signal from the deck.
- The shuttle dial lets you control the speed of the deck. To use the shuttle dial, use the mouse to point to the black notch on the dial. Press the left mouse button and keep it pressed while you turn the dial clockwise to increase the speed or counterclockwise to decrease it.

Note: The straight up position of the dial is zero or pause.

Recording From the Deck to a Clip

Before using the media server to record video from a deck you must do the folowing:

- Connect the output of the deck to the input of the video card on the server.
- Connect the audio output of the deck to the audio inputs of the server.

To record from the deck to a *clip*, follow these steps:

1. Use the Media Server Control Panel to do the following:
   - Create a new clip. (See “Creating a New Clip” on page 83 for information about how to create a new clip.)
   - Change the cue in-point, if needed.

     Note: The cue in-point in the control panel is the point in the clip to which you want the material captured.

   - If the Deck Control window is not displayed, choose View > Deck Control Panel in the menu bar of the control panel.
2. Use the Deck Control window to do the following:
   - Review the clip’s *edit point*s and change them if needed.

     Note: The edit points in the Deck Control window identify the portion of the tape that you wish to capture and the duration of the capture.

   - To see what would be recorded without actually recording into the clip, click the *Review* button.
   - To perform the edit, click the *Capture* button.

Recording From a Clip to the Deck

Before using the media server to lay back video to a deck you must do the following:

- Connect the input of the deck to the output of the video card on the server.
- Connect the audio input of the deck to the audio outputs of the server.

To record from a clip to the deck, follow these steps:

1. Use the Media Server Control Panel to do the following:
   - Load the clip from which you want to record.
   - Change the cue in-point, if needed.

     Note: The cue in-point in the control panel is the point in the clip from which you want to record.

   - If the Deck Control window is not displayed, choose View > Deck Control Panel in the menu bar of the control panel.
2. Use the Deck Control window to do the following:
   - Review the edit points and change them if needed.

     Note: The edit points in the Deck Control window identify the portion of the tape to which you want to record and the duration of the recording.

   - To see what would be recorded without actually recording onto the tape, click the *Rehearse* button.
   - To record from the clip to the deck, click the *Lay Down* button.

Monitoring the Status of the Media Server (mcstat)

There are two ways to start the Unit Status Monitor. The quickest way is to choose Utilities > Status Monitor from the Media Server Control Panel menu bar.

To monitor the status of the server, enter the following, either from the workstation on the media server or from a workstation on which the VST tools software[2] has been installed:

```
% /usr/vtr/bin/mcstat  [-v loglevel]  [-h hostname]
```

where

- *loglevel* sets the severity level of the messages that are written on STDOUT.

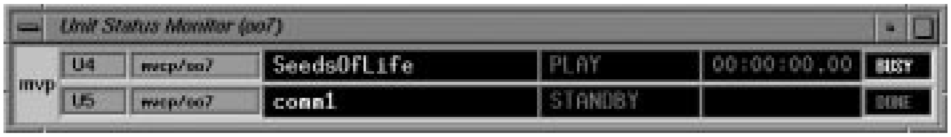
  Note: The `mcstat` program writes its log messages to the window from which it is invoked.

  If this option is omitted, all messages with a severity level of Info and above are written on STDOUT. If this option is present, *loglevel*, which can be a positive or negative number, identifies the minimum level of the messages that are written to the log. (See Table 5-2 on page 69 for the definition of the log severity levels.)

- *hostname* is the name of the media server, the default.

[2] To run the Unit Status Monitor from a remote workstation, the vst_eoe.sw32.tools subsystem must be installed on a workstation that has IRIX 6.5. For more information, see Appendix C, “Installing Video Server Technology.”

The Video Server Toolkit Unit Status Monitor appears in a separate window, which is shown in Figure 5-6. This monitor identifies each *port*, *unit*, type of connection and host, clip name (if one is loaded), current function and frame, and the status of unit. (See "About the Media Server Control Panel" on page 74 for more information.)



mvp	U4	mvcp/oo7	SeedsOfLife	PLAY	00:00:00.00	BUSY
	U5	mvcp/oo7	comm1	STANDBY		DONE

Figure 5-6 Unit Status Monitor

See "About the Media Server Control Panel" on page 74 and "About the Deck Control Window" on page 87 for a description of the status information that is displayed by the Unit Status Monitor.

Adding and Removing Clips

Clips are segments of audio and video. Managing clips involves adding and removing them to and from the SGI Media Server.

Managing clips is discussed in the following sections:

- “Media Formats and Types” on page 96
- “Adding Clips to the Media Server” on page 97
- “Transferring a Clip Segment” on page 100
- “Overriding Clip Segment Transfer” on page 100
- “Removing Clips” on page 100
- “Exporting Media Clips” on page 101

Overview of Adding Clips Procedure

This section provides a procedural overview of the tasks involved in adding clips to media server.

The following sections explain the steps in the following procedure in greater detail.

To add clips to the cache, use the following procedure.

1. Copy media to the server.

   You must use a real-time filesystem. You can either use the *`vtrutil`* tool included with VST, or a modified form of FTP, *`vtrftpd`*, installable from the VST image. You use the familiar FTP command, *`get`*, with *`vtrftpd`* to add files to the server.

   For more information, see “Adding Clips to the Media Server” on page 97.

2. Register the clip with the media server.

   Clips are registered automatically upon startup of VST.

   If you want notify the media server more quickly, or if you have added clips to the system in a different way from those listed in step 1, use one of the following options:

   - Install the *fsmon* daemon from the VST image. It periodically updates the list of clips in the cache.

     At times, however, *fsmon* may not auto detect a clip because of an unrecognized format. In that case, you must use one of the other options to notify VST.
   - Use the mvcp command, *CADD*.
   - Use the *vtrclip* tool.

   For more information see, “Notifying the Media Server” on page 99.

Media Formats and Types

The SGI Media Server supports one media type and one format.

Media Types

The media server supports intraframe media. In intraframe media, the video data for each frame is self-contained and does not depend on the data from neighboring frames. The media server supports DVCPRO media.

Media Format

The media server uses one format for storing digital media in its clip cache filesystem(s): intraframe media in DIF format—for fixed-sized frames, including DVCPRO. DIF is the 4:1 video compression frame format used by Panasonic’s DVCPRO.

Adding Clips to the Media Server

Adding clips to the media server has been partially automated. For all clip formats:

1. Transfer the media file into the filesystem, */usr/vtr/clips*.
2. For Vframe or stream format clips, transfer the index file for the clip into */usr/vtr/index*.

Transferring Clips to the Media Server

There are several ways to transfer clips into VST. Which process you use depends on the type of filesystem used. To guarantee real-time I/O rates, you must use a real-time filesystem.

To transfer clips to a real-time filesystem, use the customized version of *ftpd* (server) called *vtrftpd*. It is in subsystem *vst_eoe.sw.ftpd*. Only this version of FTP can write to real-time filesystems locally.

The feature enhancement in *vtrftpd* is used when you:

- Use FTP (client) on a system connected to the media server.
- Use FTP (client) proxy on the media server.

Both of these methods set up a TCP/IP connection with the *vtrftpd* server running on the media server.

To write to the filesystem, use of the following methods:

- On the remote archive server, use FTP's *put* command to copy a clip to the clip cache of the media server from a remote host.
- On the media server, use a proxy (secondary control) connection, *vtrftpd*, as follows to copy a clip from the remote archive system to the local clip cache:

  ```
  ftp source-system
  ftp> cd source-directory
  ftp> proxy open localhost
  ftp> proxy get filename /usr/vtr/clips/filename
  ftp> cd index
  ftp> proxy get indexname /usr/vtr/index/indexname
  ftp> proxy bye
  ftp> bye
  ```

localhost is the media server.

`vtrftpd` is not used when you only use ftp (client) on the media server because the ftpd server on another system would not be the enhanced version. To copy a clip from the archive server to VST on the media server, login to the archive system, run *`ftp`*, and use the *`put`* command.

Clip Alignment

Certain clips might need to be aligned in the filesystem for enhanced performance. In single-disk systems, media files are aligned only with the filesystem's block size so that only one I/O operation is needed to access the data.

In multi-disk, RAID, striped systems, the media files must also be aligned with the stripes.

When the media server is used to record media, it automatically places the media data in the correct, aligned locations on the disk.

Degree of Alignment

Frame-oriented media data in an intraframe clip is aligned along two boundaries:

- Minor alignment boundary
- Major alignment boundary

A single element of a frame (a video field or audio chunk) never crosses a minor or major alignment boundary. The media server will not do a read or write operation to the clip media file which crosses a major alignment boundary.

Minor Alignment

The minor alignment matches the greater of the filesystem block size, or the system memory page size. On the media server, the minor alignment is usually 16 KB, unless a larger filesystem block size was used.

Major Alignment

The major alignment matches the stripe size of the disk volume that holds the clip cache filesystem.

If the clip cache resides on a single disk, no major alignment is required.

If the clip cache resides on a single RAID subsystem, major alignment is required for efficient I/O access to the media data. Either make a real-time filesystem on the RAID and set the real-time extent size of the XFS filesystem to the desired I/O operation size, or add a configuration line to *`/usr/vtr/config/vtrfsinfo.conf`*. For example:

```
#/dev/root 2m
```

Note: *`vtrfsinfo.conf`* is not created by VST; so, you first must create the file.

If the clip cache resides on a striped XLV volume, the major alignment matches the stripe size of the XLV volume.

Notifying the Media Server

The media server detects the clips when the clip is first loaded and every time the media server starts. The clips are then listed in mcclips.

If you place a clip in the clip filesystem, the first time you attempt to access the clip through the media server, it attempts to auto-detect the clip's format and add it to the its clip list. However, until you attempt to access the clip, it will not be visible in a list of clips you obtain from the media server's Clip Manager. To see the clip immediately, use the following command:

```
# vtrclip add clipName
```

Note: There might be a delay because adding clips always has a lower priority then playout/record.

You can also use the following mvcp command to notify the media server of an added clip.

```
CADD clipName
```

However, if you install the `vst_eoe.sw_fsmon` subsystem, the media server's filesystem Monitor will monitor the clip filesystem for clips that are added or deleted and automatically update the internal clip list. Sometimes, FSmon cannot autodetect clips added to the media server when the clips require the use of an index file. In this case, you must use the *`vtrclip add`* command to add the index file and notify the media server.

Transferring a Clip Segment

`vtrftpd` honors in and out points on DIF clips for transfers from the real-time filesystem on the server. You can transfer the segment of a clip between in and out points instead of having to transfer the whole clip, saving transfer time and disk space in many cases.

If a clip has no in or out point set, *`vtrftpd`* uses these defaults:

- If the start point (beginning of the file) is invalid or missing, it is set to 00:00:00.00.
- If the in point is invalid or missing, it is set to the start point.
- If the out point is invalid or missing, it is set to the end of the file.

Overriding Clip Segment Transfer

If in and out points are set in a clip, but you want to transfer the entire clip, you can override this feature. To transfer an entire file that has in and out points set, enter the following at any point in the session:

```
ftp> site marks
```

This command is a toggle. To turn the clip segment transfer feature on again, enter the command again.

Removing Clips

There are three ways to remove clips from the media server.

- Clip Manager (mcclips)

For more information about using the Media Server Control Panel to remove a clip, see *Chapter 7, "Using Clip Manager."*

- Command line

  You can completely remove a clip from the server using the following command:

  ```
  # vtrclip rm clipName
  ```

  clipName is the name of the clip file.

- MVCP command, *CRM*.

  For more information about the *CRM* command, see Appendix H, "Multiport Video Computer Protocol (MVCP) Command Summary."

Exporting Media Clips

vtrvfutil makes the clip readable by Silicon Graphics digital media tools, such as *dminfo*, *dmconvert*, and mediaplayer, or third-party tools that use the Silicon Graphics Movie Library.

You can use the *vtrvfutil* command to add Quicktime formatting information to a clip:

```
# vtrvfutil -c makeqt clipName
```

clipName is the name of the clip file.

Note: Clip files are not necessarily usable by other tools.

Using Clip Manager

This chapter describes how to use the Clip Manager graphical user interface (GUI) to manage media *clip*s. Clip Manager is a sample GUI application that you can customize.

The following topics are discussed:

For more information about archiving clips, see Appendix G, "Archiving Clips."

Starting the Clip Manager

To start the Clip Manager, enter the following, either from the workstation connected to the media server or from a workstation on which the VST tools software[1] has been installed:

```
% /usr/vtr/bin/mcclips  [-v loglevel]  [hostname]
```

where

- *loglevel* sets the severity level of the messages that are written on STDOUT.

[1] To run the Clip Manager from a remote workstation, the vst_eoe.sw32.tools subsystem must be installed on a workstation that has IRIX 6.5. See , "Installing Video Server Technology."

Table 7-1 shows the log severity levels and codes, which are listed in decreasing order of severity

Table 7-1 Log Severity Levels

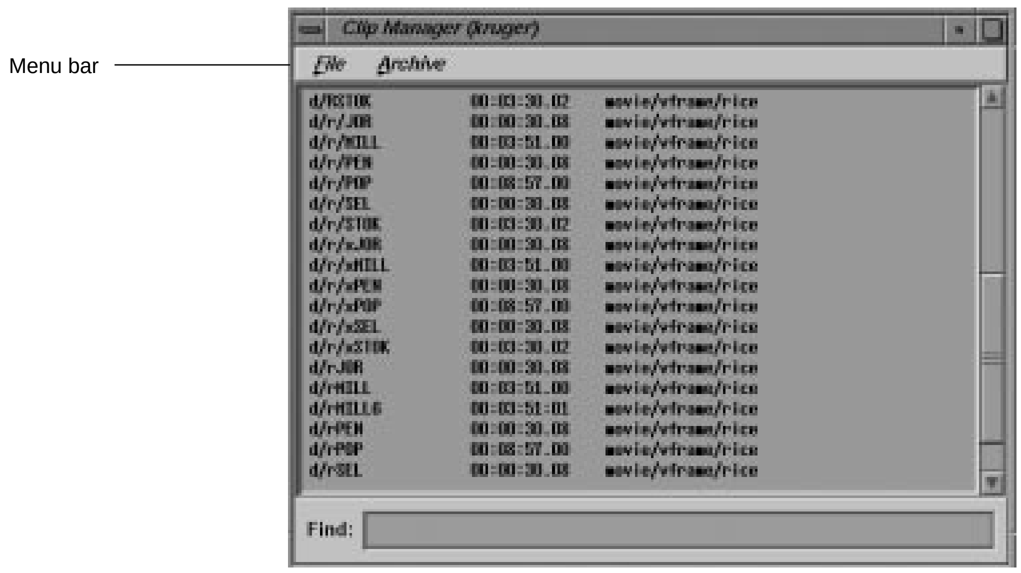
Priority	Log Level	Description
Emergency	-6	Panic condition
Alert	-5	Condition that should be corrected immediately, such as a corrupted system file
Critical	-4	Critical condition that has system-wide impact, such as a hard device error; immediate action required
Error	-3	Problem that needs correcting but does not require immediate action
Warning	-2	Possible problem but could be a transient problem that corrects itself
Notice	-1	Condition that might require attention, but is not an error condition
Info	0	Informational message
Debug*n*	*n*	Informational message that normally is of use to engineers for debugging; may be Debug1, Debug2, or Debug3, with Debug3 producing the most debugging information

Note: The `mcclips` program writes its log messages to the window from which it is invoked.

If the **-v** option is omitted, all messages with a severity level of Info and above are written to the log. If this option is present, *loglevel*, which can be a positive or negative number, identifies the minimum level of the messages that are written to the log.

- *hostname* is the name of the media server, the default.

The Clip Manager is displayed in a separate window, which is shown in Figure 7-1. The window shows the clips that are on the server. For each clip, it shows the name, the duration, and the format. (If the duration contains an "*" it means that no material has been recorded in the clip.)



```
d/RSTOK        00:03:30.02    movie/vframe/rice
d/r/JOR        00:00:30.08    movie/vframe/rice
d/r/MILL       00:03:51.00    movie/vframe/rice
d/r/PEN        00:00:30.08    movie/vframe/rice
d/r/POP        00:08:57.00    movie/vframe/rice
d/r/SEL        00:00:30.08    movie/vframe/rice
d/r/STOK       00:03:30.02    movie/vframe/rice
d/r/xJOR       00:00:30.08    movie/vframe/rice
d/r/xMILL      00:03:51.00    movie/vframe/rice
d/r/xPEN       00:00:30.08    movie/vframe/rice
d/r/xPOP       00:08:57.00    movie/vframe/rice
d/r/xSEL       00:00:30.08    movie/vframe/rice
d/r/xSTOK      00:03:30.02    movie/vframe/rice
d/rJOR         00:00:30.08    movie/vframe/rice
d/rMILL        00:03:51.00    movie/vframe/rice
d/rMILLG       00:03:51:01    movie/vframe/rice
d/rPEN         00:00:30.08    movie/vframe/rice
d/rPOP         00:08:57.00    movie/vframe/rice
d/rSEL         00:00:30.08    movie/vframe/rice
```

Figure 7-1 Clip Manager Window

Note: You can also start the Clip Manager by choosing Utilities > Clip Manager in the Media Server Control Panel menu bar.

Clip Manager Menus

The File pulldown menu in the menu bar of the Clip Manager window provides the following options:

- Info, to obtain information about a *clip*
- Rename, to rename a clip
- Delete, to delete a clip

- Protect, to set the protection levels for a clip
- Close, to close the Clip Manager window
- Exit, to close the Clip Manager window and exit the program
- Load, to load a clip into a unit controlled by Sony protocols

Except for the Close and Exit options, you must select a clip before requesting one of the options. You can select the clip by pointing to it with the mouse cursor and pressing the left mouse button.

You can also use the Find field to select a clip by entering any number of characters that match the values in either the name, the duration, or the format. For example, if you enter “xxx” in the Find field, the Clip Manager highlights the first clip it finds that contains “xxx.” If you add “y” to the Find field, it searches for a clip that contains “xxxy,” starting with the current clip.

Regardless of the method for selecting a clip, the selected clip becomes highlighted.

Note: You can access the clip-related functions in the File pulldown menu by selecting a clip and then pressing the right mouse button. You can also access the Info option by double-clicking the clip in the Clip Manager window.

The Archive pulldown menu in the menu bar provides the following options:

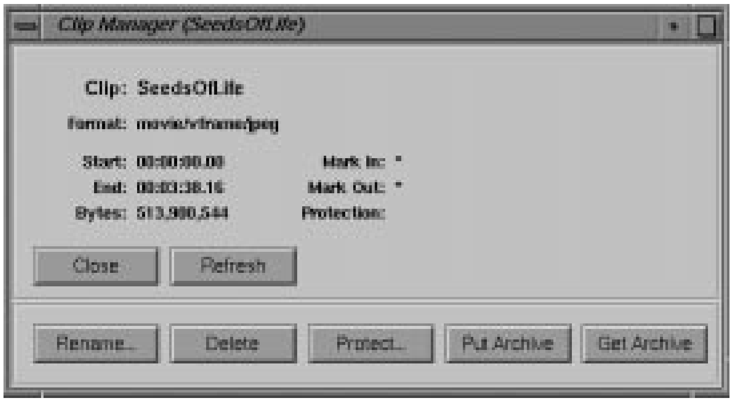
- Find, to get information about a clip in the StudioCentral 2.0 archive system
- Get New, to bring a new clip from the StudioCentral 2.0 archive system into VST
- Get, update an existing clip from the StudioCentral 2.0 archive system
- Put, to write a clip to the StudioCentral 2.0 archive system

Note: You can access the functions in this option menu by pressing the right mouse button and then selecting the Archive option in the pop up menu.

Obtaining Information About a Clip

To obtain information about a *clip*, follow these steps:

1. Select the clip in the Clip Manager window.
2. Choose File > Info from the menu bar. The clip information window, as shown in Figure 7-2, appears.



Clip: SeedsOfLife
Format: movie/vframe/jpeg
Start: 00:00:00.00 Mark In: *
End: 00:03:38.16 Mark Out: *
Bytes: 513,900,544 Protection:

Figure 7-2 The Clip Information Window

This window shows the in- and out-points of the clip, the edit points, the size of the clip, and the protection. An asterisk (*) indicates that the value is not set. For example, Figure 7-2 indicates that the edit points are not set.

The remaining sections in this chapter describe in detail the use of the buttons in this window.

3. To refresh the information in this window, click the *Refresh* button.

Renaming a Clip

To rename a clip, follow these steps:

1. Do one of the following:
   - Select the clip in the Clip Manager window and then choose File > Rename from the menu bar.
   - If the clip information window for the clip is displayed, click the *Rename...* button in the information window.

   The Rename Clip window, as shown in Figure 7-3, appears.

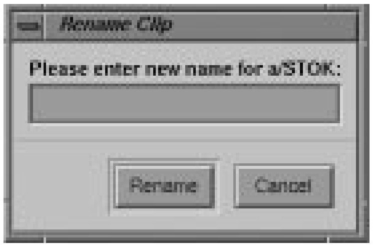


Figure 7-3 Rename Clip Window

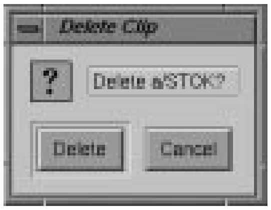
2. Enter the new name of the clip.
3. Click the *Rename* button.

Deleting a Clip

To delete a clip, follow these steps:

1. Do one of the following:
   - Select the clip in the Clip Manager window and then choose File > Delete from the menu bar.
   - If the clip information window for the clip is displayed, click the *Delete* button in the information window.

   The Delete Clip window, as shown in Figure 7-4, appears.



Delete a/STOK?

Figure 7-4 Delete Clip Window

2. Click the *Delete* button to delete the clip.

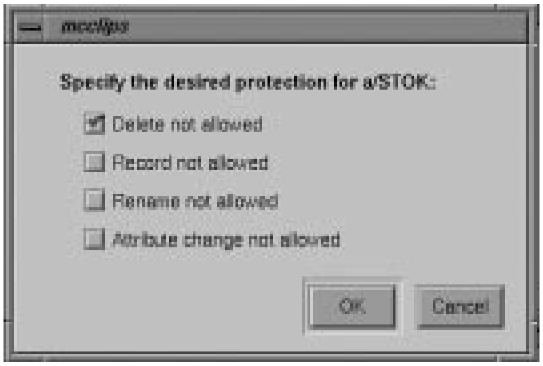
Setting the Protections for a Clip

By default, a clip can be deleted, recorded into, and renamed, and its *edit point* attributes can be changed. Therefore, if you want a clip to be protected from any of these changes, you must set the protections for that clip. This would be especially useful when multiple applications and/or *automation controller*s are using the media server.

To set the protections for a clip, follow these steps:

1. Do one of the following:
   - Select the clip in the Clip Manager window and then choose File > Protect from the menu bar.
   - If the clip information window for the clip is displayed, click the *Protect...* button in the information window.

   The Set Protections window, as shown in Figure 7-5, appears.



mcclips

Specify the desired protection for a/STOK:

- [✓] Delete not allowed
- [ ] Record not allowed
- [ ] Rename not allowed
- [ ] Attribute change not allowed

Figure 7-5 Set Protections Window

This window identifies the protections that can be set. Each protection is preceded by a check box. If there's a red check in the check box, the corresponding protection is selected. If there is no red check, the corresponding protection is not selected.

Note: The default is that a clip has no protections set. Therefore, you must set protections if you want the clip to have any.

---

2. Click the appropriate check boxes to indicate the protections that you want for this clip.

   Note: Each check box acts like a toggle switch. Click the check box once to select that protection. Click the check box a second time to deselect it.

3. Click the *OK* button to have the protections take effect.

Virtual Clips

A virtual clip (vclip) does not contain media; a virtual clip is a list of in and out points that refer to one or more clip files. A virtual clip is similar in concept to an Edit Decision List or a Play List. When you play a virtual clip, you play all of the segments of all of the clips referenced by the virtual clip, as shown in Figure 8-1.

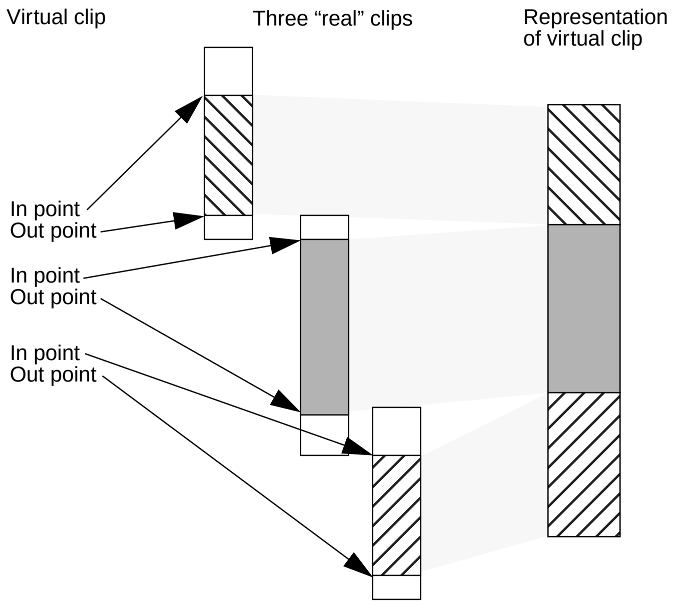


Figure 8-1 Virtual Clips

A virtual clip can also point at other virtual clips, which, in turn, point at clip files.

The remainder of the chapter explains the MVCP commands you use to manipulate vclips:

- “Virtual Clip Command Overview” on page 114
- “File Operations On Virtual Clips” on page 115
- “Working With Segments” on page 116
- “Working with Frames” on page 121

Virtual Clip Command Overview

Table 8-1 provides a summary of the MVCP commands used to manipulate virtual clips. The remainder of the chapter discusses these commands in more detail.

Table 8-1 MVCP Commands for Virtual Clips

Command	Description
CMK	Create a virtual clip.
COPN	Open a clip.
CSAV	Save a clip.
CCLS	Close a clip.
CUPS	Update clip segments.
CSLS	List clip segments.
CSRM	Remove clip segments.
CSCL	Clear clip segments.
CFNW	Insert new frames in a clip.
CFRM	Remove frames from a clip.
CFCL	Clear frames from a clip.

The first four commands are file operations for clips. The middle four operations pertain only to segmented clips, that is, virtual clips. The last three commands work on frames in virtual and non-virtual clips.

Segments

A segment is a portion of a larger clip. For example, in Figure 8-1, the virtual clip is composed of three segments. A segmented format allows a clip to be composed of segments. Currently, the only segmented format is the virtual clip (vclip).

File Operations On Virtual Clips

This section describes the MVCP commands you use to create, open, save, close, and remove virtual clips.

Creating and Opening Virtual Clips

To create a virtual clip or to open an existing one, use the following MVCP commands, respectively:

```
CMK clipName “movie/vclip”
COPN clipName
```

- *clipName* is the name of the clip to create or open. (*COPN* opens virtual and non-virtual clips.)
- *formatName* must specify a segmented format. The only segmented format supported at this time is “movie / vclip”.

Created Vclips

A clip created with *CMK* is not visible in the clip cache nor usable in the system until you add segments to it using the *CUPS* command, as described in “Adding Segments to Vclips” on page 116.

Saving, Closing, and Deleting Virtual Clips

To save, close, or delete a virtual clip, use the following MVCP commands, respectively:

```
CSAV [clipName]
CCLS [clipName]
```

clipName is the name of the clip to save or close. If *clipName* is not specified, the most recently created or opened clip is saved or closed.

Note: `CSAV`, `CCLS, and CRM` work with virtual and non-virtual clips.

Working With Segments

The MVCP commands in this section work only with segment-formatted clips; currently the only format supported is “movie / vclip”.

Once you create or open a vclip, you add segments to or remove segments from it to revise the vclip.

Adding Segments to Vclips

The `CUPS` (update) command, defined as follows, adds segments to vclips:

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

- *clip*—is the name of the vclip to which you add a segment
- *src-op*—specifies how the segment in the vclip is operated on. See “src-op Options” on page 118 for more information.
- *dest-op*—specifies whether the segment added to the vclip is inserted (FINS) into the vclip or overwrites (FOVR) segments already in the vclip. Valid values are FINS and FOVR.
- *trk-mask*—must be an asterisk (*).
- *in*—is the in point in the vclip where the segment is to be added. If this information is absent, the new segment is appended to the end of the vclip.

- *out*—is the out point in the vclip where the added segment ends.
- *src-clip*—is the name of the clip the segment is coming from.
- *src-trk-mask*—must be an asterisk (*).
- *src-in*—is the in point in the clip where the segment is coming from. If this argument is unspecified, *src-in* is set to the in point of the source clip.
- *src-out*—is the out point in the clip where the segment is coming from. If this argument is unspecified, src-out is calculated from:
  - the src-in plus in / out duration, if the in / out duration is specified
  - the out point of the source clip, if the in / out duration is not specified

Figure 8-2 illustrates some of these arguments.

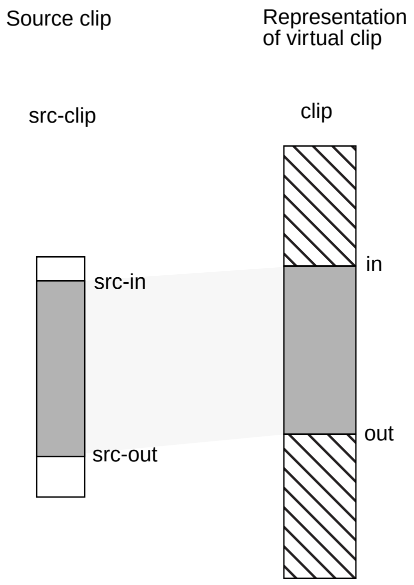


Figure 8-2 CUPS Arguments

The interval between *src-in* and *src-out* must equal the interval between *in* and *out*; otherwise, an error is returned. If you supply only one of these intervals, *CUPS* automatically makes the other interval equal to it.

Using the *CUPS* command repeatedly enables you to populate your virtual clip with segments.

src-op Options

The *src-op* options specify:

- Whether the segment data (the in points and out points, for example) are copied into the vclip or simply pointed at by the vclip.
- Whether or not the segment referenced in the source clip is erased.

The valid values for *src-op* are:

- FCP—copy the segment data (the in and out points, for example) from the source clip to the target vclip, as shown in Figure 8-3.

  The source clip must be segmented. If the source clip changes, the target vclip is unaffected, which is not the case with the FLN option.
- FLN—link the segment data in the source clip to the vclip, as shown in Figure 8-3.

  Unlike the FCP option, the source clip does not have to be segmented and if the source clip changes, its changes affect the target vclip.
- FRM—copy one or more segments in the source clip to the target vclip and then remove the copied segments from the source clip.

  The source clip must be segmented.
- FCL—copy one or more segments in the source clip to the target vclip and then clear the copied segments from the source clip.

  The source clip must be segmented. Clearing the segments, in effect, makes them black in the source clip.

When a segment from a source clip is removed (FRM), the segments before and after the copied segment are joined. When a segment is cleared (FCL), the segment remains in the source clip but it is made black, as shown in Figure 8-4.

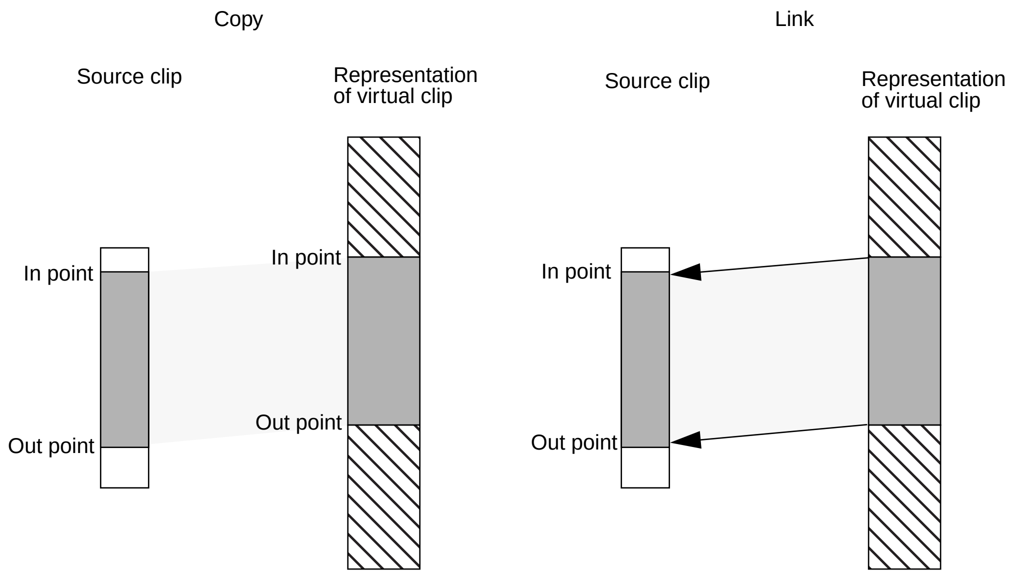


Figure 8-3 Copying vs. Linking; Where Segment Data is Stored

Listing Segments in a Vclip

To display pertinent information about the segments in a vclip, use the *CSLS* command:

```
CSLS [ clip [ track-mask [ in [ out ]]]]
```

- *clip*—is the name of the clip you are interested in.

  If *clip* is not specified, the most recently opened or created clip is used.

- *track-mask*—must be an asterisk (*).
- *in*—displays information about segments beginning after this (timecode) point in the vclip.

  If *in* is not specified, the information displayed begins with the first segment in the vclip.

- *out*—displays information about segments before this (timecode) point in the vclip.

  If *out* is not specified, the information displayed ends with the last segment in the vclip.

Use the *in* and *out* arguments to display information about segments in a subsection of the vclip.

Output

For each segment in a vclip, `CSLS` displays a line of information of the form:

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

- trk—is always an asterisk (*)
- in—the in point of the segment in the vclip
- out—the out point of the segment in the vclip
- clip—the system-dependent name of a clip created automatically in the system used for reference counting. (This argument should be ignored.)
- src-trk—is always an asterisk (*)
- src-clip—the name of the clip in the clip cache where the segment comes from
- src-in—the in point of the segment in the source clip
- src-out—the out point of the segment in the source clip

Clearing and Removing Segments in Vclips

To clear or remove segments in a virtual clip, use the following MVCP commands, respectively:

```
CSCL clip track-mask timecode
CSRM clip track-mask timecode
```

- *clip*—is the name of the clip containing the segment to be cleared or removed.

  If *clip* is not specified, the most recently created or opened clip is used.
- *track-mask*—must be an asterisk (*)
- *timecode*—is the timecode of the first frame of the segment you want cleared or removed

When a segment is cleared (`CSCL`), it remains in the specified clip but its video is made black and its audio is removed. When a segment from a specified clip is removed (`CSRM`), the segments before and after the specified segment appear contiguous, as shown in Figure 8-4.

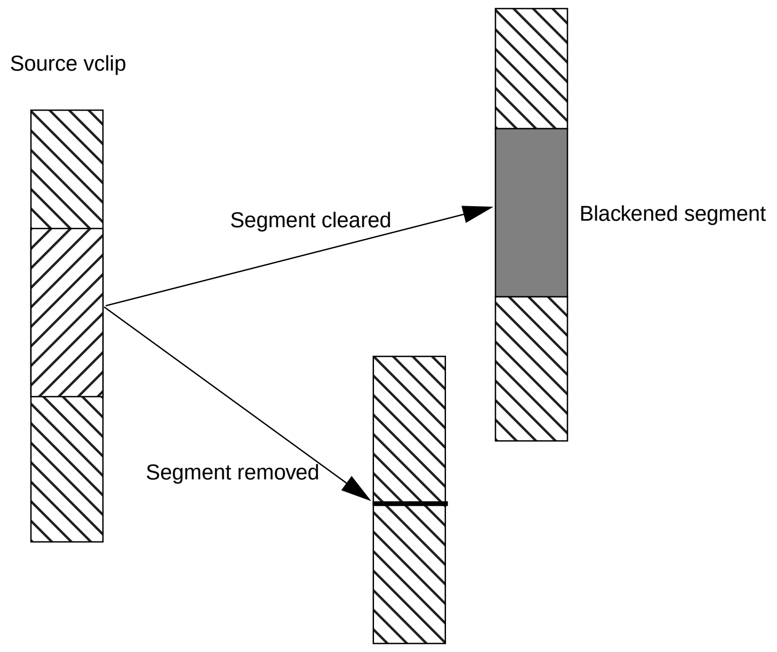


Figure 8-4 Clearing vs. Removing Segments

Working with Frames

Instead of adding or removing segments, you can use MVCP commands to add or remove frames in a vclip.

Inserting Empty Frames Into a Vclip

The `CFNW` command, defined as follows, inserts empty (black and silent) frames into a clip:

```
CFNW clip track-mask in out
```

- *clip*—is the name of the vclip into which you want to add one or more empty frames
- *track-mask*—must be an asterisk (*)

- *in*—is the in point in timecode in the vclip where the empty frames are to be inserted
- *out*—is the out point in timecode in the vclip where the inserted empty frames end

To use CFNW, the specified clip must have previously been opened or created using *COPN* or *CMK*, respectively, during the current MVCP session.

Clearing and Removing Frames in a Vclip

To clear or remove frames in a virtual clip, use the following MVCP commands, respectively:

```
CFCL clip track-mask in out
CFRM clip track-mask in out
```

- *clip*—is the name of the vclip from which you want to remove frames
- *track-mask*—must be an asterisk (*).
- *in*—is the in point in timecode in the vclip where the deletion is to start.
- *out*—is the out point in timecode in the vclip where the deletion ends.

When frames are cleared (*C*FCL), the frames remain in the specified clip but they are emptied (made black and silent). When the frames from a specified clip are removed (*C*FRM), the frames before and after the specified frames appear contiguous.

These commands are similar to FNW, FCLR, and FRM except CRCL, CFRM do not deal with units.

See Figure 8-4 for an illustration of clearing and removing.

Configuring and Using External Devices

Installation of the SGI Media Server includes configuring it to work with external devices:

- Devices that control the media server:
  - video tape recorder edit controller or other device for controlling VTRs, using a Sony-compatible (P2-compatible) RS-422 control protocol
  - automation controller using the Louth Video Disk Communications Protocol
- Devices that the media server controls: most broadcast VTRs using the Sony-compatible RS-422 control protocol
- Devices through which media are played or recorded:
  - SGI DIVO-DVC video option board: 8- and 10-bit Rec. 601 digital video playback and recording with optional 2:1 lossless data reduction or DVCPRO25 compression
  - SGI Digital Audio Option board for 24-bit uncompressed AES or ADAT digital audio
- Auxiliary devices: house time-code readers

This chapter explains installing and configuring these devices, in the following sections:

- "Configuring and Using the Sony (P2) Protocol" on page 124
- "Configuring the Media Server to Control a VTR" on page 128
- "Configuring the Server for Control by the Louth Video Disk Communications Protocol" on page 130
- "Using House Time-Code Readers With the Server" on page 133

Configuring and Using the Sony (P2) Protocol

The Sony / P2 protocol is partially supported by Video Server Technology (VST) so that the server can be controlled by standard VTR controllers. The configuration information makes the server behave similar to a video tape deck.

Note: P2 is an alternate name for the Sony protocol. For the remainder of the chapter, the term "Sony" represents both Sony and P2.

The following sections describe how to configure the srver to work with the Sony protocol:

- "Configuring Sony Protocol Control Ports" on page 125
- "Setting the Edit Controller Delay Time" on page 127
- "Changing the Clip Loaded in a Sony Controlled Logical Unit" on page 127

Note: The O2 workstation does not support deck emulation.

Caution: Opening a unit owned by a Sony port using MVCP (for example, UOPN) causes unpredictable behavior and is unsupported.

The media server does not currently support the entire Sony protocol specification. It does not support the following:

- Video output parameter selection
- Audio output parameter selection
- Audio split editing
- Insert editing for anything less than all video/audio tracks
- +/-15% playback/recording/editing

Note: VAR_FORWARD and VAR_REVERSE are fully implemented.

Configuring Sony Protocol Control Ports

Follow these steps:

1. As root, use the following command to verify that the *vst_eoe.sw.sony* subsystem is installed:

   ```
   # /usr/vtr/bin/vtrswinfo -subsys -short
    Installed software:
      vst_eoe (1265452500)
        vst_eoe.man.base (1265452500)
        vst_eoe.man.ftpd (1265452500)
        vst_eoe.man.relnotes (1265452500)
        vst_eoe.man.tools (1265452500)
        vst_eoe.sw.base (1265452500)
        vst_eoe.sw.divo (1265452500)
        vst_eoe.sw.fsmon (1265452500)
        vst_eoe.sw.ftpd (1265452500)
        vst_eoe.sw.sony (1265452500)
        vst_eoe.sw.tools (1265452500)
   ```

   If necessary, install the Sony subsystem as explained in Appendix C, “Installing Video Server Technology.”

2. Cable the controlling device to the server. For more information about cabling an RS-232 device to a server, see Appendix I.

3. For each Sony device controlled by the server, enter a line in */usr/vtr/config/control-in.conf*. See “Configuring the SGI Media Server” in Chapter 3 for instructions.

4. Edit the control defaults file for each Sony control port as needed.

   The directory */usr/vtr/config/system_defaults* contains files specifying the default control settings for system-wide resources, including the control processors managing any Sony protocol control ports.

   - Put control settings that apply to all Sony control ports in *vtr*.
   - Put control settings that are limited to specific control ports in *vtr_port*, where *port* is the server serial port number to which the controlling device is attached; for example, *vtr_1*, *vtr_2*, and so on.

   The following are controls supported by the Sony control module:

   - vtr.control.device_type_id
   - vtr.control.output.idle_mode
   - vtr.control.timecode.mode
   - vtr.control.clip.name
   - vtr.control.superimpose.enabled
   - vtr.control.edit.delay
   - vtr.control.ee.delay
   - vtr.control.ee.mode
   - vtr.control.ee.record_select
   - vtr.edit.preroll
   - vtr.edit.postroll

   See the vst-controls(5) man page for more information concerning the use and default values of these controls.

   An application can change these controls dynamically with the MVCP *SSET* command; they can also be queried with the MVCP *SGET* command. For example:

   ```
   SSET vtr_1 vtr.control.clip.name newclip
   ```

   This example creates the clip newclip, if it does not already exist, and loads it onto the units controlled by the device attached to serial port 1.

   Note: The setting for vtr.control.output.idle_mode overrides the setting for vtr.media.output.idle_mode of the underlying device.

5. Make sure that the EDIT-ON / EE-ON delay time for your edit controller matches the delay time set by VST controls. For more information, see “Setting the Edit Controller Delay Time” on page 127.

Setting the Edit Controller Delay Time

The DIVO-DVC output port processing has a finite delay that constrains how quickly the server can respond to control input. To maintain frame-accurate control, the server and an external VTR edit controller must agree on how much delay occurs between the time an EDIT_ON, EDIT_OFF, FULL_EE_ON, or FULL_EE_OFF command is received and when the command takes effect.

For DVCPRO compression, the VTR edit controller must be configured to expect an 8-frame delay. As required by your edit controller, you can increase (but not decrease) the command delay by setting the vtr.control.edit.delay or vtr.control.ee.delay controls in the appropriate system defaults file. For more information about these controls, see “Video Server Technology Controls” in Chapter 3 or the vst-controls(5M) man page.

Changing the Clip Loaded in a Sony Controlled Logical Unit

The Sony deck control protocol was originally developed to control VTRs, thus it has no facility for clip management operations including the loading and unloading of clips. The control vtr.control.clip.name can be set to change the clip currently loaded.

The GUI application *mcclips* (see Chapter 7, “Using Clip Manager,” for more information on *mcclips*) enables you to load and unload clips on ports controlled by the Sony protocol. To load a clip, for example, follow these steps:

1. Select a clip.
2. Select File > Load On Port.

You can also create a new clip on a port:

1. Select a clip.
2. Select File > Create On Port.

An application can load a new or existing clip onto the units controlled by a Sony protocol device by using the MVCP *SSET* command to set the value of vtr.control.clip.name (see “Video Server Technology Controls” in Chapter 3 or vst-controls(5M)).

Configuring the Media Server to Control a VTR

The media server can frame-accurately control a VTR or VTR-like device that supports the industry-standard Sony compatible VTR RS-422 control protocol. The VTR can be controlled interactively through a user interface or through an application for frame-accurate captures and laydowns of clips. Follow these steps:

1. As root, enter the following to verify that the *vst_eoe.sw.diaquest* subsystem is installed:

   ```
   # /usr/vtr/bin/vtrswinfo -subsys -short
    Installed software:
      vst_eoe (1265452500)
        vst_eoe.man.base (1265452500)
        vst_eoe.man.ftpd (1265452500)
        vst_eoe.man.relnotes (1265452500)
        vst_eoe.man.tools (1265452500)
        vst_eoe.sw.base (1265452500)
        vst_eoe.sw.diaquest (1265452500)
        vst_eoe.sw.divo (1265452500)
        vst_eoe.sw.fsmon (1265452500)
        vst_eoe.sw.ftpd (1265452500)
        vst_eoe.sw.tools (1265452500)
   ```

   If necessary, install the Diaquest subsystem as explained in Appendix C, “Installing Video Server Technology.”

2. Cable the server to the VTR. For more information about cabling an RS-442 device to a VST server, see Appendix I.

3. For each Sony device controlled by the server, enter a line in */usr/vtr/config/control-out.conf*. See “Configuring the SGI Media Server” in Chapter 3 for instructions.

   The following example shows the control-out.conf configuration line for a VTR connected to serial port 3:

   ```
   vtr rs422 3 38400 1
   ```

4. Edit the control defaults file as needed.

   The directory */usr/vtr/config/device_defaults* contains files specifying the default control settings for VST devices, including external VTRs controlled by the server.

   - Put control settings that apply to all controlled VTRs control ports in *dq*.
   - Put control settings that are limited to a specific controlled VTR in *dq_port*, where *port* is the VST serial port number to which the VTR is attached; for example, *dq_1*, *dq_2*, and so on.

   The following are the only controls supported by the Diaquest VTR control module:

   - vtr.edit.preroll
   - vtr.edit.postroll
   - vtr.media.output.mode
   - vtr.edit.coincidence.preroll
   - vtr.media.sync_port

     For information on the controls, see "Video Server Technology Controls" in Chapter 3 or the vst-controls(5M) man page.

     To guarantee frame-accurate control, this control must be set to the name of the server video port that is connected to the VTR. The application *mcpanel* does this automatically, but other applications must ensure that the control is set correctly.

     If the same server video port is always used with the VTR, you can set the value of vtr.media.sync_port to the name of the video port in the appropriate *dq_port* device defaults file. However, if the controlled VTR might be connected to different server video ports at different times, set the control via the application using the MVCP *SET* command. For example, before performing a frame-accurate capture from the controlled VTR to the DIVO_DVC_2 video port, set this control:

     ```
     SET dqunitname MED vtr.media.sync_port DIVO_DVC_2
     ```

5. Put the VTR into remote (or slave) mode.
6. Repeat all these steps for as many VTRs as you wish to control and as are connected to the server.

Configuring the Server for Control by the Louth Video Disk Communications Protocol

The Louth Video Disk Communications Protocol (VDCP), defined by Louth Automation, provides full-featured control of the media server using RS-232, RS-422, and TCP/IP. Connected to the SGI Media Server, the Louth processor supports the following:

- back-to-back play and record (subject to restrictions imposed by the video I/O port capabilities)
- archival management
- control of multiple video (signal) ports from a single communications (control) port

Caution: Do not open any units (using MVCP `UOPN`) belonging to the Louth control protocol processor and try to control them.

This section consists of these subsections:

- "Louth Video Disk Communications Protocol" on page 130
- "Using the Louth ADC-100 Automation Controller with the Server" on page 131

Louth Video Disk Communications Protocol

Table 9-1 summarizes the VDCP commands that the server supports.

Table 9-1 VDCP Commands Supported by the Media Server

System	Immediate	Reset/Select	Sense Request
Delete From Archive	Stop	Rename ID	Open Port
Delete Protect ID	Play	Reset Std. Time	Next
UnDelete Protect ID	Record	New Copy	Last
	Still	Sort Mode	Port Status Request
	Step	Close Port	Position Request
	Continue	Select Port	Active ID Request

Table 9-1 (continued) VDCP Commands Supported by the Media Server

System	Immediate	Reset/Select	Sense Request
	Jog	Record Init	Device ID Request
	Vari. Play	Play Cue	Device Type Request
		Cue with Data	Syst. Status Request
		Delete ID	ID Liist
		Get from Archive	ID Size Request
		Clear	ID’s Added to Arch.
		Send to Archive	ID Request
		% to Signal Full	ID’s Added List
		Record Init with Data	ID’s Deleted List
		Disk Preroll	Multi Port Status Request

Note: The media server supports Deferred (Timeline) commands.

Using the Louth ADC-100 Automation Controller with the Server

An ADC-100 Louth automation controller is connected to the media server using one or two serial ports per the video port to be controlled. If you want to play and record at the same time, two serial port connections are required, unless you are using VDCP multiple-port command support.

Follow these steps to connect and configure an ADC-100 Louth automation controller:

1. Cable the automation controller to the server. For more information about cabling an RS-442 device to the media server, see Appendix I.
2. Configure the automation controller. The information in Table 9-2 specifies how to configure a Louth ADC-100 to control the server. If you are using another automation controller that uses Louth VDCP to control a video server, you might need to configure that controller in a similar way.

Configure each Louth communications port that controls one or more video ports to use the Standard Video Disk device protocol.

Table 9-2 Louth Device Parameters

Device Parameter	Setting
VIDEO INPUT PORT IN DISK	Video port number; corresponds to signal configuration line in *control-in.conf*
VIDEO OUTPUT PORT IN DISK	Video port number; corresponds to signal configuration line in *control-in.conf*
UPDATE EVENT DURATIONS FROM DISK	Enable only on one port for each server
CONFIGURE INSTANT PLAY PREROLLS	Enabled, 0 seconds 4 frames
NUMBER (=) OF FRAMES TO SEND PLAY EARLY	3 frames (DVCPRO)
ENABLE BACK TO BACK PLAY	3 frames (DVCPRO)
ENABLE BACK TO BACK RECORD	3 frames (DVCPRO)
DISK HAS ARCHIVE	Enabled if StudioCentral archive is available; otherwise disabled
BACKUP PLAY FROM ARCHIVE SUPPORTED	Disabled
CACHE RECORD DISK SERIAL COMM. PORT NUMBER	0

3. For each Louth communications port connected to a serial port, include a control port configuration line in */usr/vtr/config/control-in.conf*. See “Control Port Configuration Line” in Chapter 3 for instructions, for example, for a RS-422 connection.
4. For each video port to be controlled through VDCP, include a signal port configuration line like that described in “Control Port Configuration Line” in Chapter 3.

The following example shows the server control-in.conf file for a configuration where VDCP is used to control 4 output video ports and 2 input video ports. The Louth communications ports controlling the outputs are connected to serial ports 3 through 6 and control DIVO_DVC_0 through DIVO_DVC_3, respectively. The Louth communications ports controlling the inputs are connected to serial ports 7 and 8 and control DIVO_DVC_0 and DIVO_DVC_1, respectively.

```
louth rs422 3 38400 1 1 29.97 8
louth rs422 4 38400 1 2 29.97 8
louth rs422 5 38400 1 3 29.97 8
louth rs422 6 38400 1 4 29.97 8
louth rs422 7 38400 1 -1 29.97 8
louth rs422 8 38400 1 -2 29.97 8


signal 1 DIVO_DVC_0
signal 2 DIVO_DVC_1
signal 3 DIVO_DVC_2
signal 4 DIVO_DVC_3
```

Using House Time-Code Readers With the Server

The media server supports the Miranda Little Red and Horita PR-232 time-code readers (LTC-to-serial translators), which are RS-232 (not RS-422) serial devices. One of these devices can be attached to the server to provide a frame-accurate time-of-day reference so that server operations can be synchronized with other studio equipment or with scheduled live or downlinked feeds.

The time-of-day signal connected to the server is used as the reference for triggering timed unit commands for playback, recording, and so on. For media devices (video ports) that support it, frame-accuracy is guaranteed when time-triggered commands are executed.

The time-of-day signal is also used to slave the time-of-day maintained by the IRIX operating system. The time-of-day is typically maintained within 1 ms of the input time signal.

To install a timecode reader and configure the VST server for it, follow these steps:

1. Connect the timecode reader's serial port to the desired RS-232 port on the server.
2. Include a control port configuration line in *`/usr/vtr/config/control-in.conf`*. See "Signal Configuration Line" in Chapter 3 for more information. Note the following:
   - For timecode input, the signal port field ("signalport" in Chapter 3) of the configuration line specifies the time channel number, which must be 1 for this release of the media server.

- Both the Miranda Little Red and Horita PR-232 connect at a serial port speed of 9600 bits / sec with no parity.

The following example shows the server *control-in.conf* file for a Horita PR-232 connected to serial port 3:

```
hsip rs232 3 9600 0 1 29.97
```

This example shows a Little Red in a 625 / 50 configuration:

```
little-red rs232 3 9600 0 1 25
```

3. Add a line for each device to *control-in.conf* to configure the server to work with Horita and the Miranda Little Red time-code readers; for example:

```
hsip rs232 2 9600 0 1 29.97
little-red rs232 2 9600 0 1 29.97
```

The format of the configuration line is explained in “Configuring the Server for Control by Remote Devices (control-in.conf File)” in Chapter 3.

PART THREE

Troubleshooting III

This part of the book contains a single chapter:

Chapter 10, “Troubleshooting,” describes some likely problem scenarios you might experience while operating your media server and what to do.

Troubleshooting

This chapter lists an assortment of common problems and their solutions.

The problems include the following:

625/50 Clips Do Not Play

The default values for the controls are NTSC-specific. If you are using a 625/50 system, set the following system control in */usr/vtr/config/system-defaults/main*:

```
vtr.main.timing_standard 625
```

This control sets the timing of the system, including the DIVO-DVC cards, to 625/50.

In */usr/vtr/config/device-defaults/DIVO_DVC*, make sure the control vtr.media.video.output.timing is set to its default value, *system*.

Adding a Clip Takes Excessively Long

Adding clips always has a lower priority than playout and recording.

DIVO-DVC Is Left in Bad State

If you find that the DIVO-DVC board has been left in a bad state, for example, you cannot reset it, use the *divo_reset* command, included in the *divo.sw.diag* images, to correct the problem. For example, the DIVO-DVC reset command is in the directory, */usr/dmedia/DIVO_DVC/bin*.

If you are using a 625/50 system, see "625/50 Clips Do Not Play" on page 137.

Crash: Semaphore Limit Exceeded

If *vvtr* crashes, you might have exceeded the number of configured pollable semaphores. The log would appear as follows:

```
C 06-14:29:23.348170   1386 Failed opening semaphore file descriptor
(sems open=147): No space left on device
A 06-14:29:23.348985   1386 FATAL SYSTEM ERROR:
SYSTEM CONFIGURATION ERROR
```

To fix the problem, follow these steps:

1. Edit */var/sysgen/master.d/usema* and increase *USMAXDEVS*, for example, to 1050.
2. Rebuild the kernel (*autoconfig*) and reboot the system to activate the change.

Crash: Audio Sync Problems When Audio Is Missing

The media server may not respond after a few seconds if you do not have an audio source. If you are having audio sync problems when there is no audio, do the following:

1. Turn the audio off: in the *`/usr/vtr/config/device-defaults`* directory, edit the DIVO-DVC file and add the following:

   `vtr.media.audio.input.channels.clip “0”`

2. If you do not have an audio source, disable input audio by setting the unit control:

   `vtr.media.audio.input.port ““`

Crash: Audio Source Lacking

The media server may not respond after a few seconds if you do not have an audio source. To fix this problem, disable input audio by setting the unit control:

`vtr.media.audio.input.port ““`

Configuring for Booting From a Backup Plex

The system normally boots from the primary root plex, that is, *`root.data.0.0`*. If the primary plex becomes unavailable, you can either label the disks and swap master and slave, or you can use the following procedure to set up the system so that it can boot from the secondary root plex, for example, *`system.data.1.0`*.

1. From the System Maintenance Menu, choose Enter Command Monitor (5).

2. Display the PROM environment variables:

   ```
   >> printenv
   SystemPartition=dksc(0,1,8)
   OSLoadPartition=dksc(0,1,0)
   root=dks0d1s0
   ...
   ```

   The swap PROM environment variable (which is set below) is not displayed because it is not saved in NVRAM.

3. Reset the SystemPartition, OSLoadPartition, and root environment variables to the values of the disk partition that contains the alternate plex and the swap environment variable to have the value of the alternate swap partition. For example:

   ```
   >> setenv SystemPartition dksc(0,2,8)
   >> setenv OSLoadPartition dksc(0,2,0)
   >> setenv root dks0d2s0
   >> setenv swap /dev/dsk/dks0d2s1
   ```

4. Exit the Command Monitor and restart the system:

   ```
   >> exit
   ...
   Option? 1
   Starting up the system...
   ...
   ```

Media Server Crash Files

Inside the */usr/vtr/adm/crash* directory is a subdirectory for each program that crashes. If a program crashes, a directory is dynamically created with that program name. Inside that directory a core file is created containing the crash information.

When the media server detects a program crash, it renames the core file to *core.number*, where *number* is the next incremental number. This scheme prevents core files from overwriting each other.

Media server crash files are saved to aid SGI support personnel in diagnosing server software errors.

PART FOUR

Manual Control and System Monitoring IV

This part of the book describes the varous ways you can manually control and monitor the SGI Media Server.

Chapter 11, “Completing Common Tasks Using MVCP Commands,” describes a grab bag of common tasks you routinely perform using MVCP commands.

Completing Common Tasks Using MVCP Commands

This chapter is a grab bag of common tasks you routinely perform using MVCP commands. Each task is discussed in a tutorial fashion to introduce you to the way MVCP works. Once you master the basic tasks presented in this chapter, you can proceed to the other features offered by MVCP commands.

The tasks presented in this chapter are not sequential, but modular. There is a flow to the MVCP commands as presented; however, one command does not necessarily build on the one presented previous to it.

After completing a number of these tasks you will gain a feeling for the MVCP commands. Appendix H describes all of the MVCP commands.

Note: For an explanation of MVCP commands, see the mvcp man page.

This chapter discusses the following tasks:

- “Manual Access to Video Server Technology” on page 144
- “Creating and Deleting a Unit” on page 145
- “Loading, Creating, and Unloading a Clip” on page 146
- “Finding the Name of a Clip” on page 146
- “Setting Edit Points” on page 146
- “Cueing Decks to Play or Record” on page 147
- “Sequencing Commands” on page 147
- “Playing a Prerecorded Clip” on page 148
- “Setting and Listing Configuration Values” on page 149
- “Listing Video and Deck Control Ports” on page 150

Manual Access to Video Server Technology

Video Server Technology (VST) applications routinely open a TCP/IP connection to a host running VST on the MVCP port, normally 5250. The application sends MVCP commands to control VST.

The port value is set in the file */usr/vtr/config/control-in.conf*. For example, to set the port to 5250, use the following line in the file:

```
mvcp tcp 5250
```

Note: Do not change the port value from 5250 unless it is absolutely necessary.

You can, however, manually control VST by opening a telnet connection to a host running VST and then issuing MVCP commands to control it. For example:

```
152% telnet server 5250
Trying 130.62.156.178...
Connected to server.
Escape character is ‘^]’.

100 VTR Ready
```

Concluding a Session

To exit the telnet session, type the following command:

```
BYE
```

Creating and Deleting a Unit

A unit is a virtual VTR. It can play and record video and audio just like a VTR.

Units can be created using the following command:

```
UADD DIVO_DVC_0 * SHAR
```

The UADD command returns the name of the unit, *unitName*, for example:

```
UADD DIVO_DVC_0 * SHAR
202 OK
U1
```

U1, in this case, controls the DIVO_0 video board, which includes both an input video port and an output video port.

For an explanation of the UADD options, see “UADD *owner* *port* *mode* *port-physical-name*” on page 248. For more information about deleting a unit, see “UCLS [*unit-name*]” on page 275.

Deleting a Unit

Units can be deleted using the following command:

```
UCLS unitName
```

Multiple Connections to a Unit

Once a unit has been created, it can be controlled by a number of MVCP connections by opening it:

```
UOPN unitName
```

When multiple MVCP connections are made to one unit, the unit is not deleted until all of the connections have deleted it.

Warning: Opening a unit owned by a Sony or Louth port using MVCP commands (for example, UOPN) causes unpredictable behavior.

Loading, Creating, and Unloading a Clip

To load and unload a clip into a unit, use the following commands, respectively:

```
LOAD unitName clipName OUT CRTE
UNLD unitName
```

unitName is the name of the unit on which the clip is loaded.

The OUT option means that the clip, *clipName*, can only be played, not recorded onto. Other valid values include IN, which means the clip will be recorded onto, and BOTH, which means the clip can be played or recorded onto.

CRTE creates the clip. Without a CRTE argument, the clip does not exist in the system.

Finding the Name of a Clip

If you do not know the name of a clip, you can list all of the clips by issuing the clip list command:

```
CLS
```

Setting Edit Points

The IN and OUT edit points specify where the source video is to begin and end playing, and where the beginning and ending recording points are on the record deck, usually the media server.

When a clip is played, by default the edit IN point is used as the starting point for PLAY.

To set edit points, use the *CEDP* command:

```
CEDP clipName inPoint outPoint
```

Note: The media outside of the in and out points is not actually erased; its just not played.

Cueing Decks to Play or Record

You can play and record on units without cueing them. There is, however, a good chance that the beginning of the playing or recording will not be clean: the frame count might be off by several frames or the audio might be garbled.

You can avoid these problems by cueing the units for playing or recording.

To cue a clip for playback, enter:

`CUE` *unitName*

To ca clip for recording, enter:

`CUER` *unitName*

Optional Arguments

The following optional CUE and CUER arguments specify the segment of media to play, the direction of play and how many times the segment is played:

```
CUE unitName <in-point> <out-point> <direction> <number-of-passes>
```

Values for *direction* include FWD (forward), BWK (backward), F/B (play forward and then play backwards), and B/F (play backward and then play forward).

Sequencing Commands

You can sequence more than one command to execute on a unit by using the append-sequential specification:

`/APP /SEQ` *command unitName*

This command specifies that the command, *command*, is appended (/APP) to the current list of commands queued for the unit, *unitName*, and the command will be executed after the previous command in the list completes (/SEQ). The alternative to waiting for completion is to preempt the previous command on the list using the flag, /IMM.

You can also do the following:

- Prepend a command in the list of commands by using /PRE instead of /APP.
- Delete the command list and replace it with a command using /FLS (flush) instead of /APP.

For more information about command sequencing, see “Command Sequencing” on page 266.

Playing a Prerecorded Clip

To play a clip, use the following procedure:

1. Open a TCP/IP connection to a host machine running VST.

   If successful, VST responds:

   ```
   100 VTR Ready
   ```

2. Create a unit, which is a virtual videotape deck:

   ```
   UADD DIVO_0 * SHAR
   ```

   For an explanation of the UADD options, see “UADD *owner* *port* *mode* *port-physical-name*” on page 248.

3. Load the clip:

   `LOAD` *unitName clipName* `OUT`

   unitName is the name of the unit on which the clip is loaded.

   The OUT option means that the clip, *clipName*, can only be played, not recorded onto.

4. Cue the clip for playback:

   `CUE` *unitName*

5. Play the clip:

   `PLAY` *unitName speed*

speed is the speed of the playback, for example, 1000 is normal play, -1000 is normal reverse play.

You can also play the clip in other modes, such as:

- `FF`—fast forward.
- `JOG`—advance or reverse one frame at a time.
- `SHTL`—(shuttle) is a variable speed fast forward or fast reverse.

For more information about the arguments for each of these commands, see "Unit Commands" on page 263.

Setting and Listing Configuration Values

VST configuration variables are called *controls*. The two major categories of controls are the following:

Device-specific—for which you use the SET and GET commands to specify and retrieve the control values for individual units.

System-wide—for which you use the SSET and SGET commands to specify and retrieve the control values for global, system values, for example, the log level.

There are two subsets of device-specific controls:

- Storage—for reading digital data off a disk or writing the data to a disk.
- Media—for communicating with the video port.

To set or get the configuration variables for a specific unit, you have to select the subset of controls you want to set by using STOR (storage) or MED (media). For example:

```
SET unitName MED vtr.media.video.input.compression.type DVCPRO
GET unitName MED vtr.media.video.output.*
```

These commands set the video compression type to DVCPRO, and return all of the video output controls for the specified unit.

To list all of the storage controls, use the asterisk (*) wildcard: GET unitName STOR *

For more information about the SET command, see “SET *unit-name* *dev-type* *control1-name* *control1-value* ...” on page 274. For more information about the SSET command, see “SSET *subsystem-name* *control1-name* *control1-value* ...” on page 260

For more information about the GET command, see “GET *unit-name* *dev-type* *control-name-pattern* ...” on page 271. For more information about the SGET command, see “SGET *subsystem-pattern* *control-name-pattern*” on page 259.

Listing Video and Deck Control Ports

To display the configuration of video and deck control ports, use the port list command:

```
PLS
```

Identifying the Audio Ports to Use

To identify which audio port to use by a given DIVO-DVC unit, use the SET command for a unit:

```
SET <unit> MED vtr.media.audio.input.port RAD1.AESIn
SET <unit> MED vtr.media.audio.output.port RAD1.AESOut
```

Alternately, you can configure a unit in */usr/vtr/config/device-defaults/**. For example, in DIVO_DVC_1 you would enter:

```
vtr.media.audio.input.port RAD1.AESIn
vtr.media.audio.output.port RAD1.AESOut
```

Configuring Audio Recording

To record audio using the media server, you must set the following controls:

- Bits in audio sampling, 24 by default—vtr.media.audio.input.sample.width
- Sampling rate, 48,000 by default—vtr.media.audio.input.rate
- Audio port used—vtr.media.audio.input.port

The audio port can either be VideoIn, if the audio is embedded in the video signal, or the name of an audio port, for example, AnalogIn or AESIn.

For information about setting control values, see “Setting and Listing Configuration Values” on page 149.

Configuring Video Recording Compression

To record video using the media server, you must set the compression type using *`vtr.media.video.input.compression.type`*. Valid value:

```
DVCPRO — for high-quality DCT-based compression
```

For information about setting control values, see “Setting and Listing Configuration Values” on page 149.

There are other commands that you might set according to your media format. For a list of those settings, see Appendix E, “4:2:2:4 Sampled Video.”

Recording a Clip

To record a clip, use the following procedure:

1. Open a TCP/IP connection to the media server.

   If successful, the server responds:

   ```
   100 VTR Ready
   ```

2. Create a unit, which is a virtual videotape deck:

   ```
   UADD DIVO_0 * SHAR
   ```

   For an explanation of the UADD options, see “UADD *media-port-name* *storage-port-name* *port-sharing-mode* [*owner-info*]” on page 246.

3. Configure VST for audio and video recording, as described in “Configuring Audio Recording” on page 150 and “Configuring Video Recording Compression” on page 151.

4. Load the clip:

   `LOAD` *unitName clipName* `BOTH CRTE`

   unitName is the name of the unit on which the clip is loaded.

The BOTH option means that the clip, *clipName*, can be played or recorded onto. The CRTE option means that the clip should be created if it does not exist.

5. Cue the clip for recording:

   `CUER` *unitName*

6. Record onto the clip:

   `REC` *unitName*

Editing Clips

You can edit clips on two levels of granularity:

- Clip
- Frame

Editing Clips

The following MVCP commands provide basic editing functionality for a clip already loaded in a unit:

- CEDP—sets in and out edit points.

  ```
  CEDIT InPoint OutPoint
  ```

- CRM—removes a clip.

  ```
  CRM clipName
  ```

- CCP—copies a clip.

  ```
  CCP clipName newClipName
  ```

- CMV—renames a clip.

  ```
  CMV clipName newClipName
  ```

- CLN—creates a new clip that shares its contents with an existing clip.

  ```
  CLN clipName newClipName
  ```

When using CLN, changing one clip also changes the renamed clip because of their link.

For more information about setting in and out editing points, see "Setting Edit Points" on page 146.

Editing Frames

The following MVCP commands manipulate the frames within a clip that is already loaded in a unit:

- FRM—remove a frame. In film, the equivalent is cutting out frames of film and splicing the two parts of the film back together.

  ```
  FRM unitName inFrame outFrame
  ```

- FCLR—changes (clears) a frame to black but does not remove it.

  ```
  FCLR unitName inFrame outFrame
  ```

- FNEW—inserts a black frame into the clip.

  ```
  FNEW unitName inFrame outFrame
  ```

- FOVR—moves frames from one part of a clip and overwrites frames in another part of the clip.

  ```
  FOVR unitName sourceIn sourceOut destIn
  ```

- FINS—moves frames from one part of a clip and inserts them in another part of the clip.

  ```
  FINS unitName sourceIn sourceOut destIn
  ```

Displaying Your Logo

You can use the MVCP *SET* command with the following controls to display your company logo, or any image, when the video port is not playing a clip:

```
vtr.media.output.idle_mode image
vtr.media.video.output.image.type user
vtr.media.video.output.image.name <name of image>
```

Image files go in */usr/vtr/data/images*.

Playing Clips from a Playlist

The commands to execute a playlist can be performed over a single MVCP control connection. Some control implementations may find it easier to use a unique MVCP control connection for each unit. The commands and the order in which they are sent is identical in either case.

1. Create a unit on the media port.

   ```
   UADD port
   202 OK
   U1
   ```

2. Create a second unit on the same media port.

   ```
   UADD port
   202 OK
   U2
   ```

3. Load the first clip into U1.

   ```
   LOAD U1 clip1 OUT
   202 OK
   clip1 movie/dif/dvcpro 108969984 108969984 00:00:00.00
    00:00:30.08 * * 29.97 19990415T004204.435814Z CL
   ```

4. Cue U1 with a mode that will return when the command is placed in the command queue.

   ```
   /SEQA CUE U1
   200 OK
   ```

5. Play U1 with a synchronization mode where a response will be returned only when the unit has started executing the command.

   ```
   /SEQA /SYNR PLAY U1
   200 OK
   ```

6. Load next clip into U2.

   ```
   LOAD U2 clip2 OUT
   202 OK
   clip2 movie/dif/dvcpro 108969984 108969984 00:00:00.00
    00:00:30.08 * * 29.97 19990415T004130.833005Z CL
   ```

7. Cue U2 with a queued command.

   ```
   /SEQA CUE U2
   200 OK
   ```

8. Play U2 as with U1 above.

```
/SEQA /SYNR PLAY U2
200 OK
```

9. Load next clip into U1.

```
LOAD U1 clip3 OUT
202 OK
clip3 movie/dif/dvcpro 108969984 108969984 00:00:00.00
 00:00:30.08 * * 29.97 19990415T004148.636950Z CL
```

10. Cue U1 with a queued command.

```
/SEQA CUE U1
200 OK
```

11. Play U1 as above.

```
/SEQA /SYNR PLAY U1
200 OK
```

Monitoring Unit State

When executing a play list, it is common to also desire display of the state of the list execution. Using MVCP this can be accomplished via polling, but a more efficient solution is to utilize unit monitoring. Unit monitoring provides asynchronous events describing the change in state of a unit, such a execution state, loaded clips, or timecode location. Details of unit monitoring can be found in the MVCP documentation for the MON command. Typically a separate MVCP connection will be initiated for monitoring.

In the example play list algorithm above, a unit monitor could be started after the units are created to trace unit execution. This would allow a control application to trace which clip was being played and the timecode location of the unit in the clip. Note that the information provided by unit monitoring describes the unit state and locations. The actual video frame timecode leaving the server port will be some fixed number of frames behind that time due to codec delays. This delay is format dependent and can be found in the table below. The table shows the difference between the time reported by the unit and the actual video frame at the server port at a that time. For instance, with DVCPRO on record the unit status reports 3 frames behind the frame on the input of the server, and on play the unit status reports 7 frames ahead of the frame on the server output port.

Table 11-1 Time Reported by the Unit and the Actual Video Frame

Format	Record Unit State (frames)	Play Unit State (frames)
DVCPRO	-3	+7

Monitoring the System

The media server provides the following ways to monitor the system:

- Status of units
- Continuous monitoring of a unit
- Listing statistics
- Error reporting

The following sections explain each of these reports in further detail:

- "Status" on page 156
- "Monitoring" on page 156
- "Statistics" on page 158
- "Error Reporting" on page 159

Status

You can display the status of one unit or all units using the following commands, respectively:

```
USTA unitName
ULS
```

Monitoring

For continuous monitoring of a unit, you use the MON command:

```
MON [unitName] [/ eventType]
```

If you omit a unit name, all of the units are monitored.

You can receive notification when any of the following events occurs on a unit in the clip cache, as appropriate:

- UADD—unit added
- URM—unit removed
- UCHG—unit state changed
- UCTL—unit control changed
- UERR—unit error
- CADD—clip added
- CRM—clip removed
- CEDP—edit points changed
- CCHG—clip media changed
- CMV—clip moved
- CCHP—clip protection changed

You can add one or more of these event types to the MON command after the slash (/) that separates the unit names from the event types.

If event types are not specified, notification of UCHG, URM, and UCTL events are returned. If a unit name is not specified, notification of each UADD event is returned.

MON prevents an MVCP connection from processing any further commands. To stop monitoring, you must either close the MVCP connection or issue another command on the connection at which time monitoring is terminated.

For more information about the MON command, see “MON [*unit-name* ...] [/ *event-type* ...]” on page 248.

Statistics

You can return statistics values for one or all components in a system using the list statistics command:

```
STLS [componentName [statisticsType]]
```

componentName is the name of the component you want statistics about. If omitted, statistics for all components are returned.

statisticsType is the type of statistics you want returned. If omitted, all statistics for the specified component are returned.

If successful, the STLS command returns a line of the following form:

componentName statisticsType statisticsValue...

To see the list of available statistics, type

```
STLS * *
```

Statistics on Statistics

You can return statistical values about a component's or all component's statistics using the following command:

```
STST [componentName [statisticsType]]
```

For example, the following command calculates all the statistics for components that contain "DIVO_DVC" in their names:

```
STST *DIVO*
```

If the command is successful, a single line is returned in the following format:

values samples min max sum mean stddev

where

- *values* is the number of statistical values matching the pattern.
- *samples* is the total number of samples collected.
- *min* is the minimum value.

- *max* is the maximum value.
- *sum* is the sum of the values.
- *mean* is the mean of the values.
- *stddev* is the standard deviation of the values.

For more information about STST, see “STST [*component-pattern* [*statistic-pattern*]]” on page 260.

Error Reporting

The media server reports three types of errors:

- MVCP command syntax errors
- Controller errors
- Unit Errors

MVCP Command Syntax Errors

All MVCP commands return error responses for syntax violations, for example, entering a letter instead of a number.

Controller Errors

All controller and non-unit errors, such as errors in renaming or deleting a clip, return a generic notification that something failed. To receive more specific information about the most recent global error that occurred for a specific controller, use the ERR command:

```
ERR
```

This command returns an error code and a short description.

Unit Errors

Unit errors are errors in the execution of unit commands, such as PLAY, REC, and FF. When looking at the status of a unit, you might find it in the error state, as described in “Status” on page 156. In that case, you might like to get more information about the error.

To return unit errors, use the UERR command:

`UERR` *unitName*

This command returns an error code and a short description.

Setting Up the Hardware

This chapter describes the steps you need to take to set up your system so that Video Server Technology (VST) can run on it in the following sections:

- “Supported Hardware” on page 161
- “Hardware Installation” on page 164
- “Initial System Administration” on page 165

Supported Hardware

This section consists of the following subsections:

- “Platform” on page 161
- “Port Configuration” on page 161
- “External Devices” on page 162
- “Disk Storage Options” on page 163

Platform

VST 1.3 works on the following platform running IRIX version 6.5.6: SGI Origin200 GIGAchannel server with SGI DIVO-DVC video option board (XIO) installed .

Port Configuration

The SGI Origin 200 GIGAchannel single-module system has six PCI slots and five XIO slots; one XIO slot can be used for the 100-Base-T XIO option board, which has six serial ports.

External Devices

The medias server can interact with a variety of external devices, as shown in Figure A-1.

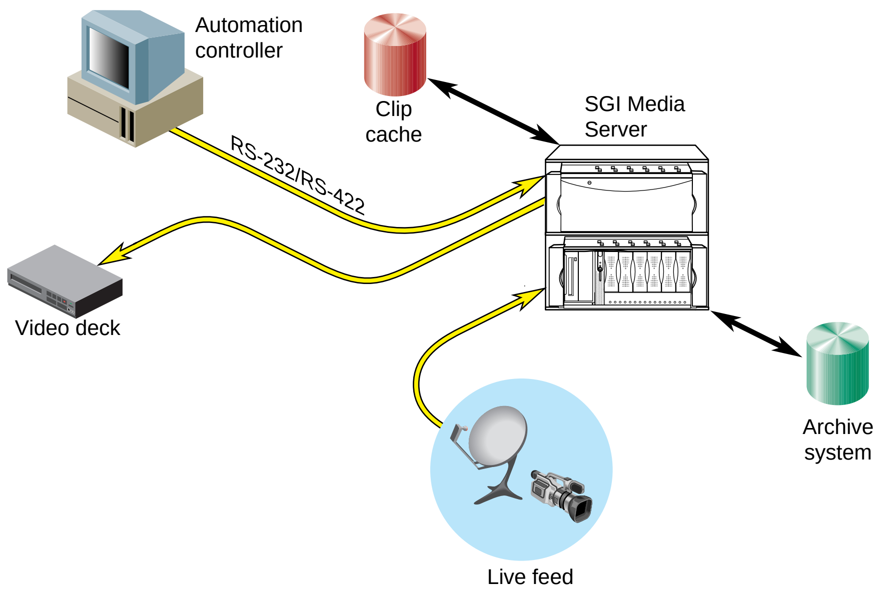


Figure A-1 SGI Media Server With External Devices

Installation of the media server includes configuring it to work with external devices:

- Devices that control the media server:
  - video tape recorder edit controller or other device for controlling VTRs, using a Sony-compatible (P2-compatible) RS-422 control protocol
  - automation controller using the Louth Video Disk Communications Protocol
- Devices that the media server controls: most broadcast VTRs using the Sony-compatible RS-422 control protocol
- Devices through which media are played or recorded:
  - SGI DIVO-DVC video option board: 8- and 10-bit Rec. 601 digital video playback and recording with optional 2:1 lossless data reduction or DVCPRO25 compression

  - SGI Digital Audio Option board for 24-bit uncompressed AES or ADAT digital audio
- Auxiliary devices: house timecode readers

Disk Storage Options

The SGI Media Server has two disk storage options:

- SGI Origin 200 GIGAchannel internal drive bays (four 18-GB Ultra SCSI disk modules (32 MB/sec))
- Ciprico 7000 Fibre Channel RAID disk array (44 MB/sec per RAID)

Bandwidth Considerations

A stream is a video/audio program. The number of streams a server can record or play at once depends on the storage option and the compression type.

For example, if you assume that a DVCPRO stream is 25 Mb/sec (3.6 MB/sec), the internal storage system can handle eight DVCPRO-compressed video streams (32 MB/3.6 MB = 8 streams).

A single Ciprico fibre channel RAID disk array has a maximum bandwidth of approximately 44 MB/sec and can support 12 DVCPRO video streams (44/3.6 = 12). However, the system limitations allow simultaneous playback and record as follows:

- 4 channel system - up to 2 record + 4 playback (6 streams total)
- 8 channel system - up to 4record + 4 playback (12 streams total

Compression Formats

The bandwidth determines the number of streams of video that can be played or recorded at once. The DVCPRO compression format requires 3.6 MB/sec.

Hardware Installation

To set up hardware for use with VST, follow these steps:

1. Install the server, interface boards, and storage following instructions in the manual(s).
2. From the system administrator at your site, obtain a name and IP address for the system.
3. Connect the system to the network using a 10-Base-T Ethernet cable between the RJ-45 connector on the Origin200 GIGAchannel master module and a network port.
4. Connect the system console (RS-232 / RS-422 serial) port on the Origin200 GIGAchannel master module to a serial port of another system or an ASCII terminal using a serial cable.

   See instructions for using an SGI workstation as the system console or attaching the system console in the *Origin200 GIGAchannel Owner's Guide.*
5. Edit */etc/uucp/Devices* to remove the comment; change

   ```
   # --A direct line so `cu -lttyd2` will work
   # Direct ttyd2 - 9600 direct
   ```

   to

   ```
   # --A direct line so `cu -lttyd2` will work
   Direct ttyd2 - 9600 direct
   ```
6. Log in to the system console using the serial port on another system. Make sure the serial port on the other system is not being used for another purpose, such as its own system console.

   Use the following command if the other system is an SGI server.

   ```
   cu -s 9600 -l ttyd2
   ```
7. Install any hardware and associated driver software related to VST, such as DIVO-DVC, following instructions in the manuals for the products and in Chapter 9, “Configuring and Using External Devices.”

Initial System Administration

Before setting up filesystems, complete the following steps to perform basic administration tasks:

1. Name the system.

   ```
   vi /etc/sys_id
   ```

2. Add system name and IP address to the hostname-address database.

   ```
   vi /etc/hosts
   ```

3. Set IP address in non-volatile RAM; also, set system up for automatic restart.

   ```
   nvram netaddr <IP Address>
   nvram AutoLoad Y
   ```

4. Set network address mask, as appropriate for the network.

   ```
   vi /etc/config/ifconfig-1.options
   ```

5. Set default system time zone.

   ```
   vi /etc/TIMEZONE
   ```

6. Set password for user root, if necessary.

   ```
   passwd
   ```

7. Reboot the system.

   ```
   reboot
   ```

Setting Up Filesystems for the SGI Media Server

This chapter describes how to set up XFS filesystems for the SGI Media Server and for the clip cache, which holds the media assets. The clip cache h is maintained on one or more filesystems across one or more disks or RAID arrays. You set up the filesystems on a system disk and a duplicate disk.

This chapter includes explicit steps for setting up the plexed filesystem. However, it is beyond the scope of this book to explain in detail the intricacies of the tools, including XFS, XLV, and Guaranteed Rate I/O (GRIO), that you use to set up your filesystem. For information about any of these tools, see the latest version of *IRIX Admin: Disks and Filesystems*.

This chapter contains these sections:

- “Preparing for Logical Volume Creation” on page 167
- “Setting Up Plexing” on page 170
- “Using GRIO” on page 187

Preparing for Logical Volume Creation

A logical volume is a set of disks or disk arrays that are treated as one disk. Logical volumes (XLVs) allow for the creation of a real-time subvolume to be used within a real-time filesystem. XLVs are also used to stripe data across a set of disks or disk arrays to form a subvolume.

This section consists of the following:

- “Using Real-Time Filesystems” on page 168
- “Determining Volume Striping Values” on page 168
- “Choosing Parameters for Creating and Mounting an XFS Filesystem” on page 169
- “Using Redundancy” on page 169

For more information about XLV, see *IRIX Admin: Disks and Filesystems*.

Using Real-Time Filesystems

An XFS filesystem can be created on an XLV disk volume that contains a real-time subvolume. The real-time subvolume is a separate data area, usually created on a separate set of disks or disk arrays, that is optimized for real-time recording and playback of digital media. When an XFS filesystem is part of the clip cache, the real-time subvolume, if available, is where the media data is stored. Only real-time filesystems have guaranteed real-time I/O rates (GRIO).

Data can be read normally from a real-time filesystem, but standard utilities, such as *ftp* and *cp*, cannot be used to write the data. However, the VST installation replaces the standard *ftp* daemon in the configuration file */etc/inetd.conf* with *vtrftpd*, an enhanced version that is installed from *vst_eoe.sw.ftpd*. Thus, invoking *ftp* automatically initiates *vtrftpd*, which supports a site marks facility that keeps track of in and out points in clips.

For more information on the use of real-time filesystems, see"Disk Storage" in Chapter 1 .

Determining Volume Striping Values

Part of the job of creating a logical volume is to supply XLV with values for striping the disks. Striping is the process of distributing media data across several disks so that different parts of the media can be simultaneously accessed, which provides a higher overall bandwidth.

Striping is highly recommended for real-time performance. You can stripe disks for the real-time subvolume of real-time filesystems (recommended), or for non-real time filesystems (not recommended).

Follow these guidelines for striping volumes:

- Choose a larger stripe unit to improve performance, but do not exceed the optimal I/O size for the individual disk devices being striped. Exceeding the optimal size does not significantly improve performance, but increases system memory requirements.

- For non-RAID disks and disk arrays, the stripe unit per disk should be at lest 256 KB ({512 disk blocks), preferably 612 KB (1024 disk blocks).
- For a 4+1 RAID-3 disk array, the stripe unit should be at least 1 MB ({2048 disk blocks), preferably 2 MB (4096 disk blocks).
- For an 8+1 RAID-3 disk array, the stripe unit should be at least 2 MB (4096 disk blocks), preferably 4 MB {(8192 disk blocks).

• If a real-time subvolume is being striped, the real-time extent size of the XFS filesystem must match the stripe size given by the following formula:

  num-striped-disks `* stripe-unit * 512bytes/block`

Choosing Parameters for Creating and Mounting an XFS Filesystem

When you create an XFS filesystem, follow these guidelines:

- Set filesystem block size to 16 KB only.
- Set the real-time extent size to the optimal I/O size for the real-time subvolume.

  If the subvolume is striped, it must match the stripe size. If the subvolume is not striped (single disk or RAID array), it should match the optimal I/O size for the disk system.

  For a single RAID, use 2 to 4 MB for the extent size. For more information about determining the optimal I/O size for a disk, see the striping guidelines in “Using GRIO” on page 187.

For more information about XFS, see *IRIX Admin: Disks and Filesystems*.

Using Redundancy

You can set up one or more servers for clip mirroring, to reflect the state of the primary server’s clip cache. Appendix F, “Configuring a Server for Clip Mirroring,” describes how to configure for these options.

Setting Up Plexing

Setting up plexing consists of several procedures, explained in the following subsections:

- "Partitioning the System Disk" on page 170
- "Installing System Software on the System Disk" on page 174
- "Duplicating the System Disk" on page 177
- "Installing an XFS Flexlm XLV License" on page 178
- "Making the Root Partition on the System Disk Into an XLV Volume" on page 179
- "Creating the Plex for the Root" on page 181
- "Partitioning the Clip Cache" on page 183
- "Creating the XLV Volume for the Clip Cache" on page 185

Caution: Partitioning the disk destroys all data on it.

Partitioning the System Disk

Table B-1 summarizes the desired setup for the system disk in a redundant system.

Table B-1 System Disk Partitions

Partition	Contents	Size
0	root, type xlv	256 MB
1	swap, type raw	512 MB
6	usr, type xlv	rest of disk
7	data subvolume, type xlv	1024 MB
8	volhdr	2 MB

Note: Have ready the following CDs: IRIX 6.5.6 and the IRIX 6.5 CDs: Installation Tools, Foundation 1, Foundation 2, Applications, and NFS.

This subsection gives instructions for partitioning the system disk and breaking the current IRIX installation. Follow these steps:

1. Perform initial system administration, as explained in “Initial System Administration” on page 165 in Appendix A.
2. Back up data on the disk as necessary.
3. Enter

   ```
   # fx -x
   ```

   The system responds:

   ```
   fx version 6.5, Apr 13, 1999
   fx: "device-name" = (dksc)
   ```

4. Press **Enter**. The system responds:

   ```
   fx: ctlr# = (0)
   ```

5. Press **Enter**. The system responds:

   ```
   fx: drive# = (1)
   ```

6. Press **Enter**. The system responds:

   ```
   fx: lun# = (0)
   ```

7. Press **Enter**. The system responds:

   ```
   ...opening dksc(0,1,0)
   ...drive selftest...OK
   Scsi drive type == SGI     IBM  DGHS18Y    0190

   ----- please choose one (? for help, .. to quit this menu)-----
   [exi]t               [d]ebug/             [l]abel/             [a]uto
   [b]adblock/          [exe]rcise/          [r]epartition/
   fx>
   ```

8. Enter **repartition/expert**. The system responds:

   ```
   Warning: you will need to re-install all software and restore user
   data from backups after changing the partition layout.  Changing
   partitions will cause all data on the drive to be lost.  Be sure you
   have the drive backed up if it contains any user data.  Continue?
   ```

9. Enter **yes**. The system responds:

   ```
   Enter .. when done
   fx/repartition/expert: change partition = (0)
   ```

10. Press **`Enter`**. The system responds:

    ```
    before:  type xfs        block       0,         0 MB
                             len:        0 blks,    0 MB
    fx/repartition/expert: partition type = (xfs)
    ```

11. Enter **`xlv`**. The system responds:

    ```
    fx/repartition/expert: base in megabytes = (0)
    ```

12. Enter **`2`**. The system responds:

    ```
    fx/repartition/expert: size in megabytes (max 17363) = (0)
    ```

13. Enter **`256`**. The system responds:

    ```
     after:  type xlv        block    4096,         2 MB
                             len:   524288 blks,  256 MB
    fx/repartition/expert: change partition = (1)
    ```

14. Enter **`1`**. The system responds:

    ```
    before:  type xfs        block       0,         0 MB
                             len:        0 blks,    0 MB
    fx/repartition/expert: partition type = (xfs)
    ```

15. Enter **`raw`**. The system responds:

    ```
    fx/repartition/expert: base in megabytes = (0)
    ```

16. Enter **`258`**. The system responds:

    ```
    fx/repartition/expert: size in megabytes (max 17107) = (0)
    ```

17. Enter **`512`**. The system responds:

    ```
    after:  type raw        block  528384,       258 MB
                             len:  1048576 blks,  512 MB
    fx/repartition/expert: change partition = (6)
    ```

18. Enter **`7`**. The system responds:

    ```
    before:  type xfs        block    4096,         2 MB
                             len:  35558944 blks, 17363 MB
    fx/repartition/expert: partition type = (xfs)
    ```

19. Enter **`xlv`**. The system responds:

    ```
    fx/repartition/expert: base in megabytes = (2)
    ```

20. Enter **`770`**. The system responds:

    ```
    fx/repartition/expert: size in megabytes (max 16595) = (17363)
    ```

21. Enter **1024**. The system responds:

```
 after:  type xlv        block 1576960,       770 MB
                         len:  2097152 blks, 1024 MB
fx/repartition/expert: change partition = (8)
```

22. Enter **6**. The system responds:

```
before:  type xfs        block       0,         0 MB
                         len:        0 blks,    0 MB
fx/repartition/expert: partition type = (xfs)
```

23. Enter **xlv**. The system responds:

```
fx/repartition/expert: base in megabytes = (0)
```

24. Enter **1794**. The system responds:

```
fx/repartition/expert: size in megabytes (max 15571) = (0)
```

25. Enter **15570** (that is, max -1) The system responds:

```
after:  type xlv        block 3674112,      1794 MB
                         len:  31887360 blks, 15570 MB
fx/repartition/expert: change partition = (7)
```

26. Enter **..** (two periods). The system responds:

```
----- partitions-----
part  type        blocks            Megabytes   (base+size)
  0: xlv       4096 + 524288          2 + 256
  1: raw     528384 + 1048576       258 + 512
  6: xlv    3674112 + 31887360     1794 + 15570
  7: xlv    1576960 + 2097152       770 + 1024
  8: volhdr       0 + 4096            0 + 2
 10: volume       0 + 35563040        0 + 17365

capacity is 35563040 blocks

----- please choose one (? for help, .. to quit this menu)-----
[ro]otdrive          [o]ptiondrive        [e]xpert
[u]srrootdrive       [re]size
fx/repartition>

* Enter /label/sync

writing label info to dksc(0,1,0)

----- partitions-----
part  type        blocks            Megabytes   (base+size)
```

```
  0: xlv        4096 + 524288         2 + 256
  1: raw      528384 + 1048576      258 + 512
  6: xlv     3674112 + 31887360    1794 + 15570
  7: xlv     1576960 + 2097152      770 + 1024
  8: volhdr        0 + 4096           0 + 2
 10: volume        0 + 35563040       0 + 17365

capacity is 35563040 blocks

----- please choose one (? for help, .. to quit this menu)-----
[ro]otdrive           [o]ptiondrive         [e]xpert
[u]srrootdrive        [re]size
fx/repartition>
```

27. Enter **`/exit`**.

Installing System Software on the System Disk

When the label is written to disk, the system must be powered on again. Follow these steps:

1. Power on the system. After the PROM executes, a message like the following appears:

```
Autoboot failed.
dksc(0,1,0)unix: no such file or directory.

Hit Enter to continue.
```

2. Press **`Enter`**. The System Maintenance Menu appears:

```
 System Maintenance Menu

1) Start System
2) Install System Software
3) Run Diagnostics
4) Recover System
5) Enter Command Monitor
```

3. Choose “Install System Software” (2). The following message appears:

```
Installing System Software...

Press <Esc> to return to the menu.
1) Remote Directory  X) Local CD-ROM
```

```
Enter 1-2 to select source type, <esc> to quit,
or <enter> to start:
```

4. Load the IRIX 6.5.6 CD into the remote or local CD-ROM drive.
   - If you are installing from a local CD-ROM, enter **`2`** and continue at step 7.
   - If you are installing from a remote directory, enter **`1`**. The system responds:

     ```
     Enter the name of the remote host:
     ```
5. If you are installing from a remote CD-ROM, enter the name of the remote host. The system responds:

   ```
   Enter the remote directory:
   ```
6. Enter **`/CDROM/dist`**. The system responds:

   ```
   1)[Remote Directory]  X) Local CD-ROM
         *a) Remote directory /CDROM/dist from server remoteHost.

   Enter 1-2 to select source type, a to select the source, <esc> to
   quit, or <enter> to start:
   ```
7. Press **`Enter`**. The system responds:

   ```
   Copying installation program to disk.
   ......... 10% ......... 20% ......... 30% ......... 40% ....... 50%
   ......... 60% ......... 70% ......... 80% ......... 90% ....... 100%

   Copy complete
   IRIX Release 6.5 IP27 Version 04151556 System V - 64 Bit
   Copyright 1987-1999 Silicon Graphics, Inc.
   All Rights Reserved.

   Setting rbaud to 19200
   WARNING: Cannot load runtime symbol table from bootp'ed kernel.
            Loadable modules will not be registered or loaded.
   Creating miniroot devices, please wait...

   Current system date is Mon May 17 15:30:19 PDT 1999

   Mounting file systems:

       /dev/miniroot           on  /
       /dev/dsk/dks0d3s0       on  /root
       /dev/dsk/dks0d3s6       on  /root/usr
   ```

```
Invoking software installation.
```

8. The miniroot prompts for hostname, network address, and netmask; enter these for your system.
9. At the Inst prompt, enter **admin** to display the Admin menu.
10. At the Admin prompt, enter **sh** to get a shell.
11. In the shell, enter

    ```
    # mkfs /dev/dsk/dks0d3s6
    ```

    The system responds:

    ```
    meta-data=/dev/dsk/dks0d3s6   isize=256   agcount=16, agsize=249133
    blks
    data     =                    bsize=4096  blocks=3986116, imaxpct=25
             =                    sunit=0     swidth=0 blks, unwritten=1
    naming   =version 1           bsize=4096
    log      =internal log        bsize=4096  blocks=1000
    realtime =none                extsz=65536  blocks=0, rtextents=
    ```

12. Enter

    ```
    # mount /dev/dsk/dks0d3s6 /root/usr
    # exit
    ```

13. The Admin menu appears; enter **exit** to get to the Inst main menu.
14. Insert the IRIX 6.5 CDs in the following order:
    - Installation Tools
    - Foundation-1
    - Foundation-2
    - Applications
    - NFS

    From the Inst menu, select Open to open the distributions on each CD. Unselect the distributions that you do not want.
15. Install the following non-default subsystems:

    ```
    Inst> install eoe.sw.xlv eoe.sw.xfsrt eoe.sw.xlvplex
    Inst> go
    ```

    Installing the 6.5 subsystems takes a bit of time.

16. Replace the last IRIX 6.5 CD with the IRIX 6.5.6 CD. Select the maintenance stream.

    ```
    Inst> keep N
    Inst> go
    ```

17. Enter the following to edit *root/etc/fstab*:

    ```
    Inst> Admin
    Admin> sh
    # cat > /root/etc/fstab
    /dev/root /  xfs  rw,raw=/dev/rroot 0 0
    /dev/usr /usr  xfs  rw,raw=/dev/rusr 0 0
    ```

18. Enter **Crtrl+D** to close *root/etc/fstab*. Exit from the shell and Admin, and quit Inst.

19. The system prompts for a reboot; enter **yes** at the prompt.

20. Perform an initial system administration as explained in “Initial System Administration” on page 165 in Appendix A.

Duplicating the System Disk

Once you have installed the system software on the system disk, duplicate it; follow these steps:

1. Choose another disk as a backup option disk (for example, disk 2) and enter:

    ```
    # fx -x "dksc"(0,2)
    ```

2. Follow steps 8 through 27 from the earlier section “Partitioning the System Disk” on page 170.

3. As root, create a filesystem on partition 0:

   ```
   # mkfs /dev/dsk/dks0d2s0
   ```

   The system responds:

   ```
   meta-data=/dev/dsk/dks0d3s0      isize=256    agcount=8, agsize=8192
   blks
   data     =                       bsize=4096   blocks=65536,
   imaxpct=25
            =                       sunit=0      swidth=0 blks,
   unwritten=1
   naming   =version 1              bsize=4096
   log      =internal log           bsize=4096   blocks=1000
   realtime =none                   extsz=65536  blocks=0, rtextents=0
   ```

4. Create filesystem on */dev/dsk/dks0d2s6* and */dev/dsk/dks0d2s7* in a similar way.

5. Use *xfs_copy* to copy the current system disk to the duplicate disk, with root copied into */dev/dsk/dks0d2s0* and */usr* copied into */dev/dsk/dks0d2s6*:

   ```
   xfs_copy /dev/dsk/dks0d1s0 /dev/dsk/dks0d2s0
   xfs_copy /dev/dsk/dks0d1s6 /dev/dsk/dks0d2s6
   ```

   Ignore messages about the mounted source drive.

Installing an XFS Flexlm XLV License

Once you have duplicated the system disk, use the following procedure to set up the plexing:

1. To make sure that plexing is enabled and supported, start *xlv* and run the *show config* command:

   ```
   # xlv_mgr
   xlv_mgr> show config
   ```

   The system responds:

   ```
   Plexing license: present
   Plexing support: present
   ```

2. Exit *xlv_mgr*:

   ```
   xlv_mgr> exit
   ```

3. If a plexing license is not present, to install the license in *`/var/flexlm/license.dat`*. Provide a symlink to the *`/etc/flexlm`* directory:

   ```
   # ln -s /var/flexlm /etc/flexlm
   ```

4. If you do not have a license for XLV, you must request one through the Key-O-Matic Web page http://www.sgi.com/Support/Licensing/. If your request is granted, you receive a license by e-mail or in your Web browser.
5. Once you receive the license, log on to the media server.
6. As superuser, open the license file.

   ```
   # vi /etc/flexlm/license.dat
   ```

7. Copy and paste license lines from the web site into the license file; for example:

   ```
   FEATURE XLV sgifd 2.000 01-jan-0 0 6CC5738489148F24823C \
   HOSTID=#1761963733 vendor_info="XLV XFS PLEXING" \
   ISSUER="Silicon Graphics, Inc."
   ```

8. Save the file and exit *`vi`*.

Making the Root Partition on the System Disk Into an XLV Volume

The next procedure is to make the root partition on the system disk into an XLV volume; follow these steps:

1. Enter:

   ```
   # xlv_make
   ```

   The system responds:

   ```
   xlv_make>
   ```

2. Enter **`vol root.`** The system responds:

   ```
   xlv_make>
   ```

3. Enter **`data`**.

   ```
   root.data
   xlv_make>
   ```

4. Enter **`ve -force /dev/dsk/dks0d1s0.`** The system responds:

   ```
   root.data.0.0
   xlv_make>
   ```

5. Enter **`end`**. The system responds:

   ```
   Object specification completed
   xlv_make>
   ```

6. Enter **`vol usr`**. The system responds:

   ```
   usr
   xlv_make>
   ```

7. Enter **`data`**. The system responds:

   ```
   usr.data
   xlv_make>
   ```

8. Enter **`ve -force /dev/dsk/dks0d1s6`**. The system responds:

   ```
   usr.data.0.0
   xlv_make>
   ```

9. Enter **`end`**. The system responds:

   ```
   Object specification completed
   xlv_make>
   ```

10. Enter **`exit`**. The system responds:

    ```
    Newly created objects will be written to disk.
    Is this what you want?(yes)
    ```

11. Enter **`yes`**. The system responds:

    ```
    Invoking xlv_assemble
    ```

    The result is XLV volumes named *`root`* and *`usr`* that contain the root and usr partitions on the system disk. Since XLV preserves the data in partitions, the contents of the root and usr partitions are preserved. The **-force** option to the *`ve`* command was used because a mounted partition was included in the volume.

12. Open *`/etc/fstab`* and add the following entry:

    ```
    /dev/xlv/usr/   /usr   xfs   rw   0   0
    ```

13. Create a */etc/config/xlv_plexd.options* file with the following content:

    ```
    # cat > /etc/config/xlv_plexd.options
      -b 1024 -w 50

    Press ctrl-D   # to close the file
    ```

    These commands make sure that a revive does not use too much of the disk's bandwidth.

14. Reboot the system so that it switches from running from the root partition on the system disk (*/dev/dsk/dks0d1s0*) to running from the logical volume */dev/xlv/root*.

Creating the Plex for the Root

After making the root partition on the system disk into an XLV volume, create the plex for the root. Follow these steps:

1. Create the plex for the root out of */dev/dsk/dks0d2s0*, and call it *root_plex*; also create plex for usr out of */dev/dsk/dks0d2s6*, and call it *usr_plex*.
2. Enter

   ```
   # xlv_make
   ```

   The system responds:

   ```
   xlv_make>
   ```

3. Enter **plex root_plex**. The system responds:

   ```
   root_plex
   xlv_make>
   ```

4. Enter **ve /dev/dsk/dks0d2s0**. The system responds:

   ```
   root_plex.0
   xlv_make>
   ```

5. Enter **end**. The system responds:

   ```
   Object specification completed
   xlv_make>
   ```

6. Enter **plex usr_plex**. The system responds:

   ```
   usr_plex
   xlv_make>
   ```

7. Enter **ve /dev/dsk/dks0d2s6**. The system responds:

   ```
   usr_plex.0
   xlv_make>
   ```

8. Enter **end**. The system responds:

   ```
   Object specification completed
   xlv_make>
   ```

9. Enter **exit**. The system responds:

   ```
   Newly created objects will be written to disk.
   Is this what you want?(yes)
   ```

10. Enter **yes**. The system responds:

    ```
    Invoking xlv_assemble
    ```

11. Add *sash* to the volume header of the option disk used for the *root_plex*. It enables booting from of the *root_plex* if the primary root fails.

    ```
    # dvhtool -v get sash /tmp/sash /dev/rdsk/dks0d1vh
    # dvhtool -v add /tmp/sash sash /dev/rdsk/dks0d2vh
    ```

12. Attach the *root_plex* to the root volume, and *usr_plex* to the *usr* volume using *xlv_mgr*.

    ```
    # xlv_mgr
    ```

    The system responds:

    ```
    xlv_mgr>
    ```

13. Enter the following three lines in *xlv_mgr*:

    ```
    xlv_mgr> attach plex root_plex root.data
    xlv_mgr> attach plex usr_plex usr.data
    xlv_mgr> quit
    ```

    When the shell prompt returns, the system automatically begins a plex revive so that the two plexes contain the same data.

Partitioning the Clip Cache

After creating the plex for the root, partition the clip cache. Follow these steps:

1. Determine the SCSI ID and bus info of the Ciprico FC disk arrays.

   ```
   # hinv
   ```

   The system responds:

   ```
   FPU: MIPS R10010 Floating Point Chip Revision: 0.0
   CPU: MIPS R10000 Processor Chip Revision: 2.6
   2 180 MHZ IP27 P
   ...

   Integral SCSI controller 3: Version Fibre Channel AIC-1160, revision
   2
   Disk drive: unit 1 on SCSI controller 3
   Integral SCSI controller 2: Version Fibre Channel AIC-1160, revision
   2
   Disk drive: unit 1 on SCSI controller 2
   ...

   DIVO Video: controller 2 unit 2: Input, Output
   DIVO Video: controller 3 unit 3: Input, Output
   IOC3 external interrupts: 1
   ```

2. Start *fx*:

   ```
   # fx -x
   fx: devicename = (dksc)
   ```

3. Press **Enter**. The system responds:

   ```
   fx: ctlr# = (0)
   ```

4. Enter the controller where the Ciprico device is, for example, **3**. The system responds:

   ```
   fx: drive# = (1)
   ```

5. Press **Enter**. The system responds:

   ```
   fx: lun# = (0)
   ```

6. Press **Enter**. The system responds:

   ```
   ----- please choose one (? for help, .. to quit this menu)-----
   [exi]t             [d]ebug/           [l]abel/           [a]uto
   [b]adblock/        [exe]rcise/        [r]epartition/     [f]ormat
   fx>
   ```

```
----- partitions-----
part  type       blocks            Megabytes   (base+size)
  7: xlv       4096 + 142258752       2 + 69462
  8: volhdr       0 + 4096           0 + 2
 10: volume       0 + 142262848       0 + 69464

capacity is 142262848 blocks

----- please choose one (? for help, .. to quit this menu)-----
[ro]otdrive          [o]ptiondrive         [e]xpert
[u]srrootdrive       [re]size
fx/repartition>
```

7. Enter **o** for option drive. The system responds:

   ```
   fx/repartition/optiondrive: type of data partition = (xfs)
   ```

8. Press **Enter**. The system responds:

   ```
   fx/repartition/optiondrive: create log partition? = (no)
   ```

9. Press **Enter**. The system responds:

   ```
   Warning: you will need to re-install all software and restore user
   data
   from backups after changing the partition layout.  Changing
   partitions
   will cause all data on the drive to be lost.  Be sure you have the
   drive
   backed up if it contains any user data.  Continue? yes

   ----- partitions-----
   part  type       blocks            Megabytes   (base+size)
     7: xlv       4096 + 142258752       2 + 69462
     8: volhdr       0 + 4096           0 + 2
    10: volume       0 + 142262848       0 + 69464

   capacity is 142262848 blocks

   ----- please choose one (? for help, .. to quit this menu)-----
   [ro]otdrive          [o]ptiondrive         [e]xpert
   [u]srrootdrive       [re]size
   fx/repartition>
   ```

10. Enter **/label/sync.** The system responds:

    ```
    writing label info to dksc(0,3,0)

    ----- partitions-----
    part  type       blocks            Megabytes   (base+size)
      7: xlv       4096 + 142258752       2 + 69462
      8: volhdr       0 + 4096           0 + 2
     10: volume       0 + 142262848       0 + 69464

    capacity is 142262848 blocks

    ----- please choose one (? for help, .. to quit this menu)-----
    [ro]otdrive           [o]ptiondrive         [e]xpert
    [u]srrootdrive        [re]size

    fx/repartition>
    ```

11. Enter **/exit**.
12. Repeat steps 3 through step 11 for each Ciprico RAID array.

Creating the XLV Volume for the Clip Cache

After partitioning the clip cache, create the XLV volume for it. Follow these steps:

1. Create the XLV volume for the clip cache using the data partitions on the system and option disk, and the two fibre channel RAID arrays. Enter:

   ```
   # xlv_make
   ```

   The system responds:

   ```
   xlv_make>
   ```

2. Enter the volume's name, for example, `vol vtr`. The system responds:

   ```
   vtr
   xlv_make>
   ```

3. Enter **data**. The system responds:

   ```
   vtr.data
   xlv_make>
   ```

4. Enter **plex**. The system responds:

   ```
   vtr.data.0
   xlv_make>
   ```

5. Enter **ve dks0d1s7**. The system responds:

   ```
   vtr.data.0.0
   xlv_make>
   ```

6. Enter **plex**. The system responds:

   ```
   vtr.data.1
   xlv_make>
   ```

7. Enter **ve dks0d2s7**. The system responds:

   ```
   vtr.data.1.0
   xlv_make>
   ```

8. Enter **rt**. The system responds:

   ```
   vtr.rt
   xlv_make>
   ```

9. Enter **plex**. The system responds:

   ```
   vtr.rt.0
   xlv_make>
   ```

10. Enter the following:

    ```
    xlv_make> ve -stripe -stripe_unit 4096 /dev/dsk/dks2d1s7
    /dev/dsk/dks3d1s7.
    ```

    The system responds:

    ```
    vtr.rt.0.0
    xlv_make>
    ```

11. Enter **end**. The system responds:

    ```
    Object specification completed
    xlv_make>
    ```

12. Enter **exit**. The system responds:

    ```
    Newly created objects will be written to disk.
    Is this what you want?(yes)
    ```

13. Enter **yes**. The system responds:

    ```
    Invoking xlv_assemble
    ```

14. Create an XFS filesystem on the clip’s logical volume:

    ```
    # mkfs_xfs -b size=16k -r extsize=4m /dev/xlv/vtr
    ```

15. Create an entry in the filesystem table:

```
# vi /etc/fstab
/dev/xlv/vtr  /usr/vtr/clips xfs     rw,raw=/dev/rxlv/vtr  0 0
```

16. Mount the *clips* filesystem:

```
# mount -av
```

17. Create required links for */var/adm* and */var/tmp:*

```
# mkdir /usr/vtrtmp
# rm -r /var/tmp
# rm -r /usr/tmp
# ln -s /usr/vtrtmp /var/tmp
# ln -s /var/tmp /usr/tmp
```

Using GRIO

High-performance media streaming is of utmost importance in using the media server. Guaranteed Rate I/O (GRIO) provides high-performance streaming by enabling a user application to reserve part of a system's I/O resources for its exclusive use. It is used for real-time retrieval and storage of data streams. GRIO is an installable component in the IRIX operating system.

The media server assumes that GRIO is running on real-time subvolumes. The supported configuration requires a filesystem with a real-time subvolume for clip storage and uses GRIO for bandwidth management. Other configurations can be used for development purposes or very small systems, but performance is not guaranteed.

The media server also assumes that the disks are striped according to the guidelines specified in "Determining Volume Striping Values" on page 168.

This section consists of the following subsections:

- "Determining Disk I/O Size" on page 188
- "Adjusting the Maximum Bandwidth of the SCSI Controllers" on page 188
- "Adjusting the Maximum Bandwidth of the Disks" on page 189

For more information about GRIO and its installation, see *IRIX Admin: Disks and Filesystems*.

Determining Disk I/O Size

The first step for using GRIO is deciding between 256 KB or 512 KB I/O per disk:

- Choose 256 KB for lower latency and less memory usage, with relatively reduced performance.
- Choose 512 KB for greater performance but with greater memory usage, resulting in increased latency. This choice makes sense for systems with plenty of memory.

To use GRIO, you must adjust the maximum bandwidth rating for the controllers and the disks by adding REPLACE or ADD lines to the `/etc/grio_disks` file; REPLACE is for controllers, ADD is for disks. The next sections explain adjusting maximum bandwidth for SCSI controllers and disks, respectively.

Adjusting the Maximum Bandwidth of the SCSI Controllers

To specify the bandwidth for a fibre channel SCSI controller, add to the `/etc/grio_disks` file a REPLACE line with the hardware tree path to the controller. For example:

```
REPLACE /hw/module/1/slot/io7/MotherBoard/pci/3/scsi_ctlr/0 2048K 40
REPLACE /hw/module/1/slot/io7/MotherBoard/pci/1/scsi_ctlr/0 2048K 40
```

The fibre channel SCSI controller is thus capable of 80 MB/sec each, or 40 I/Os of 2 MB/sec each. The recommended value is 80 MB/sec.

Use the command `grio -P -d` *disk_devicename* to determine the hardware graph path name for the SCSI controller. The controller is the second hardware graph name listed. For example:

```
matrix 15# grio -P -d /dev/dsk/dks0d1s7
Device Path:

/hw/module/1/slot/io1/MotherBoard/pci/0/scsi_ctlr/0/target/1/lun/0/disk
        /hw/module/1/slot/io1/MotherBoard/pci/0/scsi_ctlr/0
        /hw/module/1/slot/io1/MotherBoard/pci/controller
        /hw/module/1/slot/MotherBoard/node/xtalk/0/xbow
        /hw/module/1/slot/MotherBoard/node

vcpqa5 2# grio -P -d /dev/dsk/dks4d2s7
Device Path:
```

```
/hw/module/1/slot/io2/fibre_channel/pci/1/scsi_ctlr/0/target/2
   /lun/0/disk
/hw/module/1/slot/io2/fibre_channel/pci/1/scsi_ctlr/0
/hw/module/1/slot/io2/fibre_channel/pci/controller
/hw/module/1/slot/MotherBoard/node/xtalk/0/xbow
/hw/module/1/slot/MotherBoard/node
```

Adjusting the Maximum Bandwidth of the Disks

To adjust the maximum bandwidth of the disks, insert an ADD line into the */etc/grio_disks* file to specify disk bandwidth. Note the following:

- Disk I/O size must have the same value as the stripe unit size of the filesystem on the disk.
- The SCSI identifier must match the SCSI identifier of the attached RAID array. If the identifiers do not match, GRIO and, therefore, VST are unable to start; playout and recording from that disk are impossible.

Obtaining the RAID Array SCSI Identifier

The SCSI identifier in the ADD line must match the SCSI identifier of the attached RAID. If the identifiers do not match, GRIO and,therefore, V ST are unable to start; playout and recording from that disk is impossible.

To identify the SCSI ID of attached RAID array or other SCSI disk, follow these steps:

1. Use the *versions* command to make sure that the XFS GRIO subsystem, *eoe.sw.xfsrt*, is installed.

   ```
   % versions eoe.sw.xfsrt
   I  eoe.sw.xfsrt 02/24/97  XFS Realtime & Guaranteed-Rate Support
   ```

2. Use the *hinv* command to produce a list of the disks in the system.:

   ```
   # hinv -c disk
   Integral SCSI controller 3: Version Fibre Channel AIC-1160, revision 2
     Disk drive: unit 1 on SCSI controller 3
   Integral SCSI controller 2: Version Fibre Channel AIC-1160, revision 2
     Disk drive: unit 1 on SCSI controller 2
   Integral SCSI controller 0: Version QL1040B (rev. 2), single ended
     Disk drive: unit 1 on SCSI controller 0
     Disk drive: unit 2 on SCSI controller 0
     Disk drive: unit 3 on SCSI controller 0
   ```

```
  Disk drive: unit 4 on SCSI controller 0
  Disk drive: unit 5 on SCSI controller 0
Integral SCSI controller 1: Version QL1040B (rev. 2), single ended
```

3. Using the information in the previous step, get the SCSI ID with the following command:

   # **fx "dksc(*c*,*n*)"**

   where *c* is the SCSI controller number, and *n* is the disk number on that controller.

   For example, the root disk is controller 0 disk 1. An attached RAID should have its own controller and should be the only disk on that controller.

   In the following example, controller 2 is a fibre channel controller and disk 1 is the RAID array (the only disk on the fibre channel card). The drive type is Ciprico Rimfire 7010 03.1.

```
# fx "dksc(2,1)"
Unable to determine drive type, defaulting to SCSI, ctrlr 0, drive 1
fx version 6.5, Apr 13, 1999
...opening dksc(2,1,0)
fx: partition already in use by xlv logical volume
fx: devname            seq owner   state
fx: /dev/rdsk/dks2d1s7    1 xlv     part of xlv vol

fx: Warning:  this disk appears to have mounted filesystems.
         Don't do anything destructive, unless you are sure
         nothing is really mounted on this disk.
...drive selftest...OK
Scsi drive type == Ciprico Rimfire 7010    03.1

----- please choose one (? for help, .. to quit this menu)-----
[exi]t              [d]ebug/            [l]abel/
[b]adblock/         [exe]rcise/         [r]epartition/
fx> exit
```

4. Restart GRIO:

   # **/etc/init.d/grio stop**
   # **/etc/init.d/grio start**

   The system responds:

```
Total number of licensed grio streams is unlimited.
#
```

   When GRIO is misconfigured, it fails to start and gives a brief description of the problem.

5. When the GRIO daemon runs, check the bandwidth for the storage devices:

   ```
   # grio -C -d /dev/dsk/devname
   ```

   The system responds with the bandwidth information; for example:

   ```
   GRIO information for path of  disk device /dev/dsk/devname

   GRIO information for device:
   /hw/module/1/slot/io7/xbox_dualxtown/pci/0/scsi_ctlr/0/target/2/lun/
   0/disk
   opt i/o size:  4194304 bytes
   reservations:    1 ops currently reserved (max. per quantum),   10
   ops maximum
      bandwidth:    36864 kbytes/sec available,    40960 kbytes/sec
   maximum

   GRIO information for device:
   /hw/module/1/slot/io7/xbox_dualxtown/pci/0/scsi_ctlr/0
   opt i/o size:  4194304 bytes
   reservations:    1 ops currently reserved (max. per quantum),   25
   ops maximum
      bandwidth:    98304 kbytes/sec available,   102400 kbytes/sec
   maximum
   ```

Inserting the ADD Line for Disk Bandwidth

Insert an ADD line into the */etc/grio_disks* file to specify disk bandwidth. Set the I/O size to the same value as the stripe unit size of the filesystem on that disk.

For example, if you have a Ciprico 7000 fibre channel RAID storage system, you might use the following line:

```
ADD     “Ciprico Rimfire 7010    03.1” 2048K 22
```

This line says that the RAID array should produce 44 MB/sec, or, in this case, 22 I/Os at 2 MB/sec. (The recommended value is 44 MB/sec.) The filesystem on this disk has a stripe size of 2 MB.

You can insert multiple ADD entries for systems that have filesystems with different stripe unit sizes on the same type of disk.

Now the user application can be started. Files created on the real-time subvolume can be accessed using guaranteed-rate I/O.

Installing Video Server Technology

This chapter provides a step-by-step description of installing the Video Server Technology (VST) software in the following sections:

- "Preparing for Installation" on page 193
- "Installing VST" on page 194
- "Confirming the Installation" on page 196

Preparing for Installation

Before you install VST, make sure you have the materials required for the installation and that your workstation meets all the VST requirements. Follow these procedures:

1. To install VST, you need the IRIX 6.5.6 CD:

   Note: VST was tested for compatibility with specific hardware and software configurations. Refer to the VST release notes for the configurations supported for this release.

2. The product release notes contain the latest information for the software required to use VST, such as required patches. To read the release notes from the CD included with the product, become superuser and enter the following command:

   ```
   # /CDROM/CDgrelnotes
   ```

3. Before installing VST, install the patches required for VST, as specified in the release notes. Use Software Manager or the *inst* program to install any required patches.

4. To install VST over a network, you must be successfully connected to a network. For more information about connecting to a network, consult your system administrator.

Installing VST

Once your server is properly configured, use the following procedure for installing VST over a network or from a local CD.

1. Become a superuser by entering the following command:

   ```
   % su
   ```

2. Make sure that the *sysadmdesktop.sw* subsystem is present. This subsystem is included in the IRIX distribution.

3. Start the installation program and identify the source of the VST image, as follows:

   ```
   # inst -f servername:distribution_directory
   ```

   For example, if you are installing over the network, use a line similar to the following:

   ```
   # inst -f host1 :dist/vst/dist
   ```

   Or, if you are installing from a CD, use

   ```
   # inst -f cdHost:/CDROM/dist
   ```

   Or, if you are installing from a local CD, use

   ```
   # inst -f /CDROM/dist
   ```

4. List the subsystems to install as follows:

   ```
   Inst> l
   ```

   This command lists all the subsytems to install; Table C-1 summarizes them.

Table C-1 VST Default and Optional Subsystems

Systems and Subsystems	Default	Description
vst_eoe	Yes	Video Server Technology 1.3
vst_eoe.books which includes		SGI Media Server documentation
vst_eoe.books.MS400_800_UG.html		*T*his manual, HTML format
vst_eoe.books.MS400_800_UG.pdf		*T*his manual, PDF fomat
vst_eoe.man which includes	Yes	VST 1.3 man pages
vst_eoe.man.base		VST 1.3 base man pages
vst_eoe.man.ftpd		VST 1.3 man pages for enhanced FTP daemon
vst_eoe.man.relnotes		VST 1.3 release notes
vst_eoe.man.tools		VST 1.3 base applications man pages
vst_eoe.sw which includes		Video Server Technology 1.3 software
vst_eoe.sw.base		VST 1.3 base software
vst_eoe.sw.clipmirror		VST 1.3 clip-mirroring module
vst_eoe.sw.diaquest		VST 1.3 Diaquest deck-control module
vst_eoe.sw.divo_dvc		VST 1.3 DIVO-DVC video module
vst_eoe.sw.failsafe		VST 1.3 configuration files for IRIS FailSafe 1.2
vst_eoe.sw.fsmon	Yes	VST 1.3 File System Import Monitor (FSmon)
vst_eoe.sw.ftpd	Yes	VST 1.3 enhanced FTP daemon
vst_eoe.sw.horita		VST 1.3 Horita time-code input module

Table C-1 (continued) VST Default and Optional Subsystems

Systems and Subsystems	Default	Description
vst_eoe.sw.little-red		VST 1.3 Miranda Little Red time-code input module
vst_eoe.sw.louth		VST 1.3 Louth VDCP control module
vst_eoe.sw.sony		VST 1.3 Sony protocol module
vst_eoe.sw.studiocentral		VST 1.3 StudioCentral 2.0 archive interface
vst_eoe.sw.tools	Yes	VST base applications

5. Mark for installation any optional VST-related software you might need. For example, if you are using a DIVO-DVC board, or the Louth or Sony protocol, install one or more of the following options, as appropriate:

   ```
   Inst> i vst_eoe.sw.divo_dvc
   Inst> i vst_eoe.sw.louth
   Inst> i vst_eoe.sw.sony
   ```

6. Initiate the installation, as follows:

   ```
   Inst> go
   ```

7. Quit the installation program, as follows:

   ```
   Inst> quit
   ```

8. To run a secondary server that takes over VST operations when the primary server fails, repeat all preceding steps for the secondary server(s). Following instructions in Appendix F, “Configuring a Server for Clip Mirroring,” install IRIS FailSafe or configure the secondary server(s) for clip mirroring.

Confirming the Installation

To confirm the successful installation of VST, complete the following steps:

1. Start VST by entering the following command as a superuser:

   ```
   # /usr/vtr/bin/vtrstart
   ```

2. To see if VST started, enter

   ```
   # /usr/vtr/bin/vtrstat
   ```

(The utility *vtrstat* is described in “vtrstat” on page 64 in Chapter 4.) If VST is running, the response is as follows:

```
Video server on hostname is running.
```

3. Verify the media ports are supported by entering

   ```
   # /usr/vtr/bin/vtrstat -ports
   ```

   vtrstat lists the supported ports; for example:

   ```
   #  Port    Type    Description
   ----------------------------------------------------------
   1  DIVO_DVC_0 Video   SGI Digital Video Option (DVC)
   2  DIVO_DVC_1 Video   SGI Digital Video Option (DVC)
   3  DIVO_DVC_2 Video   SGI Digital Video Option (DVC)
   4  DIVO_DVC_3 Video   SGI Digital Video Option (DVC)
   5  DIVO_DVC_4 Video   SGI Digital Video Option (DVC)
   6  DIVO_DVC_5 Video   SGI Digital Video Option (DVC)
   7  DIVO_DVC_6 Video   SGI Digital Video Option (DVC)
   8  DIVO_DVC_7 Video   SGI Digital Video Option (DVC)
   ```

You can also confirm the installation by examining the server log for startup messages: the contents of the /*usr/vtr* and */var/adm/vtr/logs* directories:

```
# more /usr/vtr/adm/logs/vtrlog
```

Logging begins with output like the following:

```
I 22-10:46:31.484095  55938 ---- LOGGING STARTS ----
I 22-10:46:31.484885  55938 VCP-Recorder starting
I 22-10:46:35.979391  55938 System:
I 22-10:46:35.979682  55938   Platform: Origin 200 GIGAchannel
(hostname=oo7 serial=690d416b)
I 22-10:46:35.979809  55938   Processors: 2x225MHz R10000 (IP27)
I 22-10:46:35.979970  55938   Secondary Cache: 2 Mbytes
I 22-10:46:35.980128  55938   Memory: 512 Mbytes
I 22-10:46:35.980261  55938 Video devices:
I 22-10:46:35.980377  55938   DIVO_DVC_0 (module=1/GIGAchannel slot=12
serial=HTM962 part=030-1387-001E)
I 22-10:46:35.980501  55938     DVCPRO_CODEC (serial=HTM769
part=030-1388-001B)
I 22-10:46:35.980756  55938     DVCPRO_CODEC (serial=HTM802
part=030-1388-001B)
I 22-10:46:35.981005  55938   DIVO_DVC_1 (module=1/GIGAchannel slot=13
serial=HTM970 part=030-1387-001E)
I 22-10:46:35.981181  55938     DVCPRO_CODEC (serial=HTM789
part=030-1388-001B)
I 22-10:46:35.981586  55938     DVCPRO_CODEC (serial=HTM810
part=030-1388-001B)
I 22-10:46:35.981718  55938   DIVO_DVC_2 (module=1/GIGAchannel slot=14
serial=HJH608 part=030-1387-001E)
I 22-10:46:35.981844  55938     DVCPRO_CODEC (serial=HJH541
part=030-1388-001B)
I 22-10:46:35.981964  55938     DVCPRO_CODEC (serial=HJH546
part=030-1388-001B)
I 22-10:46:35.982120  55938   DIVO_DVC_3 (module=1/GIGAchannel slot=15
serial=HJH287 part=030-1387-001E)
I 22-10:46:35.982346  55938     DVCPRO_CODEC (serial=HJH644
part=030-1388-001B)
I 22-10:46:35.982479  55938     DVCPRO_CODEC (serial=HJH553
part=030-1388-001B)
I 22-10:46:35.982599  55938 Audio devices:
I 22-10:46:35.982713  55938   RAD1 (module=1/GIGAchannel pci=2)
I 22-10:46:35.982830  55938   RAD2 (module=1/O200 pci=5)
I 22-10:46:35.982948  55938   RAD3 (module=1/O200 pci=6)
I 22-10:46:35.983206  55938   RAD4 (module=1/O200 pci=7)
```

The *crash* directory contains crash files for the VST executable *vvtr*.

Table C-2 describes directories in */usr/vtr*.

Table C-2 /usr/vtr Subdirectories

Subdirectory	Description
*bin/**	Commands and the VST executable *vvtr*
*clips/**	Clip cache
*config/**	Configuration files
*data/**	Files in this directory are static data or image files used by various VST components
*images/**	The directory in which still images are created when the MediaCache VTR Control Protocol (MVCP) CIMG command is executed (this command is explained in Appendix H, “Multiport Video Computer Protocol (MVCP) Command Summary”)
install/	Files used during installation
lib/perl	Perl modules
lib32/.so*	Core and shared libraries
lib32/modules/ .so*	External interface device / control / format modules

IRIX Monitoring Tools

SGI provides a collection of monitoring tools that can be used with the media server. They include the following:

- *sar*: system activity reporter; reports operating system activity
- *gr_osview*: graphical system monitor; graphically displays real-time usage of certain system resources
- *osview*, a text version of *gr_osview*
- Performance Co-Pilot (PCP): serves as an interface for existing reporting tools, such as graphing performance data over time

Before exporting media data recorded by the media server to IRIX tools, you must manipulate the data. For more information about exporting media data, see Chapter 6, “Adding and Removing Clips.”

Using sar

sar, the System Activity Reporter, is an activity counter. The command line options allow you to specify the kinds of activities you want measured, such as disk utilization. To use *sar*, follow these steps:

1. Enable *sar* using the following command:

   ```
   # chkconfig sar on
   ```

2. Reboot your system.

   When your system reboots, the *sar* data collector starts.

3. Specify the system activity you want *sar* to measure; for example, to measure disk utilization, use a command similar to the following:

   ```
   # sar -d 2 10
   ```

The **-d** command-line option specifies the display of disk utilization. The numbers, 2 and 10, specify how often you want to take reports. In this example, 10 reports are taken every 2 seconds.

Note: For a complete list of *sar* command line options, see the sar(1M) man page.

The following is example output of *sar*.

```
vsta 100# sar -d 2 10

IRIX64 vsta 6.4 02121744 IP27    12/19/1997

11:15:10     device %busy  avque  r+w/s  blks/s    w/s wblks/s  avwait
avserv
11:15:12
dks2d1     8    1.0      3   12002      0       0     0.0    28.3
dks3d69    8    1.0      1   12257      0       0     0.0    53.3
dks0d1     8    5.4      7     120      7     120    49.3    11.3
dks1d1     0    0.0      0       0      0       0     0.0     0.0
dks0d2     0    0.0      0       0      0       0     0.0     0.0
dks1d2     0    0.0      0       0      0       0     0.0     0.0
dks0d3     0    0.0      0       0      0       0     0.0     0.0
dks1d3     0    0.0      0       0      0       0     0.0     0.0
dks0d4     0    0.0      0       0      0       0     0.0     0.0
dks1d4     0    0.0      0       0      0       0     0.0     0.0
dks0d5     0    0.0      0       0      0       0     0.0     0.0
dks1d5     0    0.0      0       0      0       0     0.0     0.0
dks0d6     0    0.0      0       0      0       0     0.0     0.0
dks1d6     0    0.0      0      16      0      16     0.0     0.0
```

Using gr_osview

gr_osview provides a graphical display of system resources use. This display provides a real-time window into the overall operation of the system, as shown in Figure D-1.

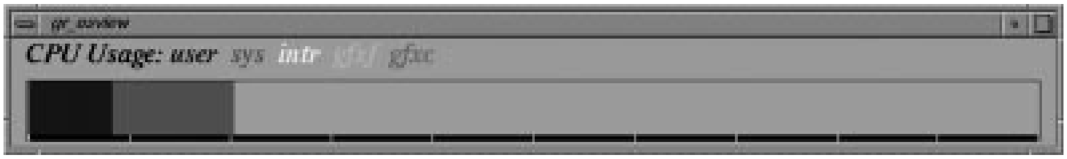


Figure D-1 gr_osview

For more information about this utility, see its man page, gr_osview(1M)

Using osview

osview is a text version of *gr_osview*. The following show example output of *osview:*

```
Osview 2.1 : One Second Average   vcpqa2      10/02/98 09:30:25 #14
int=5s
 Load Average            4MB pages    0    vidintr        0
   1 Min      0.008      16MB pages   0    drop_add       0
   5 Min      0.000  System Activity     TCP
   15 Min     0.000    syscall      206    conns          0
 CPU 0 Usage           read           3    sndtotal       0
   %user       0.00    write          0    rcvtotal       0
   %sys        1.20    fork           0    sndbyte       38
   %intr       0.00    exec           0    rcvbyte        0
   %sxbrk      0.00    readch        45  UDP
   %idle      98.80    writech       78    ipackets       1
 CPU 1 Usage           iget           0    opackets       0
   %user       0.00  Block Devices         dropped        0
   %sys        0.60    lread          0    errors         0
   %intr       0.00    bread          0  IP
   %sxbrk      0.00    %rcache      0.0    ipackets       1
   %idle      99.40    lwrite         0    opackets       0
 Wait Ratio            bwrite     22.4K    forward        0
   %IO          0.0    wcancel        0    dropped        0
   %Swap        0.0    %wcache      0.0    errors         0
   %Physio      0.0    phread         0  NetIF[ef0]
 System Memory         phwrite        0    Ipackets       2
   Phys      512.0M *Swap                  Opackets       0
    kernel    26.7M *System VM             Ierrors        0
     heap      7.7M  Memory Faults         Oerrors        0
      stream  48.0K    vfault         0    collisions     0
     zones     7.7M    protection     0 *NetIF[lo0]
     ptbl      2.9M    demand         0  Scheduler
    fs ctl     9.8M    cw             0    runq           0
    fs data  105.9M    steal          0    swapq          0
     delwri       0    onswap         0    switch        65
    free     350.5M    oncache        0 *Large page stats
    userdata  19.2M    onfile         0 *Interrupts
    pgallocs      0    freed          0 *PathName Cache
 Node[0]               unmodswap      0 *Heap
     Totalmem505.1M    unmodfile      0 *EfsAct
     freemem 350.5M    iclean         0 *XfsAct
     64k page  1.6K *TLB Actions        *Getblk
     256k pages   0  Video              *Vnodes
exit 1MB pages    0    vidioctl       0
```

Using Performance Co-Pilot (PCP)

Performance Co-Pilot (PCP) provides a suite of tools that co-operate to deliver distributed, integrated performance-monitoring and performance-management services across a spectrum of performance domains, as shown in Figure D-2.

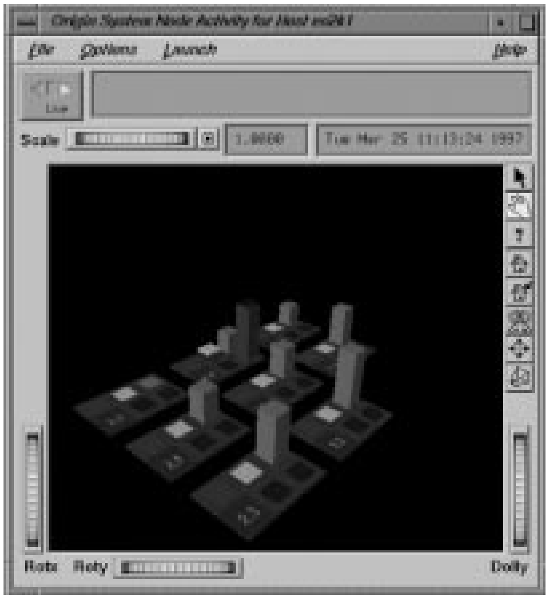


Figure D-2 Performance Co-Pilot

For more information about Performance Co-Pilot, see its man page, pcp(1M).

4:2:2:4 Sampled Video

The SGI Media Server supports recording and playout of 4:2:2:4 sampled video with alpha. The alpha channel is used as a key channel in most production environments. The media server also supports recording the 4:2:2 video and the key channel independently and later combining them into a single 4:2:2:4 clip using two DIVO-DVC boards.

Playing and Recording 4:2:2:4 sampled video is described in the following sections:

- “Setting VST Controls” on page 207
- “Workflow to Generate a Single 4:2:2:4 Clip” on page 209
- “Sample MVCP Scripts” on page 210

Setting VST Controls

Recording a specific type of media requires setting the appropriate VST controls so that the media server records in the correct mode.

Playing 4:2:2:4 video does not require setting any VST controls; VST detects 4:2:2:4 video and changes the settings based on the media type automatically.

For all of the media and clips being recorded, you generally work with CCIR601 colorspace, which is set as follows:

```
vtr.media.video.input.colorspace ccir601
```

The media server drives the DIVO-DVC board, by default, to use only the single link SDI mode (over Link A of the DIVO-DVC board). This mode works with 4:2:2 median and the key channel.

For 4;2:2:4 clips, both links (link A and link B) of the DIVO-DVC board need to be used. The DIVO-DVC needs to operate in dual-link-sdi mode (Standard Digital Interface), which is set as follows:

```
vtr.media.video.input.format sdi-dual-link
```

Controls to Set to Record 4:2:2

Set the following control to record 4:2:2:

```
vtr.media.video.input.packing R242_8
vtr.media.video.input.colorspace ccir601
vtr.media.video.input.compression.type none
vtr.media.clip.format default
vtr.media.video.input.format sdi
```

R242_8 specifies 8-bit packing format for 422 clips.

4:2:2 clips are recorded as uncompressed so that they can be edited.

Controls to Set to Record Alpha Channel

Set the following control to record alpha:

```
vtr.media.video.input.packing 4_8
vtr.media.video.input.colorspace ccir601
vtr.media.video.input.compression.type none
vtr.media.clip.format default
vtr.media.video.input.format sdi
```

4_8 specifies 8-bit packing format for alpha-only clips.

Controls to Set to Record 4:2:2:4

Set the following controls to record 4:2:2:4

```
vtr.media.video.input.format sdi-dual-link
vtr.media.video.input.packing R2424_8
vtr.media.video.input.compression.sampling 4224
vtr.media.video.input.colorspace ccir601
vtr.media.clip.format default
```

- sdi-dual-link refers to a mode using two standard digital interface (SDI) links for transport of 4224 or 4444 video.
- R242_8 specifies 8-bit packing format for 422 clips.
- 4224 specifies 4:2:2:4 sampling.

4:2:2:4 is generally compressed.

Workflow to Generate a Single 4:2:2:4 Clip

The following procedure describes how to record 4:2:2:4 sampled video with alpha using the following hardware setup shown in Figure 5-1.

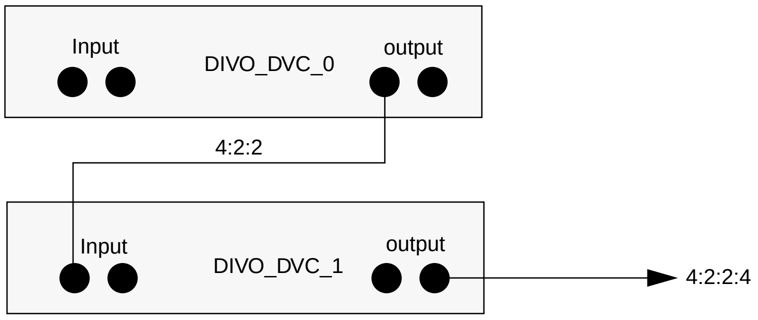


Figure 5-1 DIVO-DVC Configuration for Recording 4:2:2:4

1. Record uncompressed 4:2:2 sampled video on one SDI input of a DIVO-DVC.
2. Record the uncompressed alpha channel of the same video using one SDI input of a DIVO-DVC.
3. Use the wall clock to play the 4:2:2 and alpha synchronously through two DIVO-DVC boards where the output of one, playing the 4:2:2 video, is used as the input of the second DIVO-DVC board. Because the second DIVO-DVC board can play the alpha and record simultaneously, the second DIVO-DVC board combines the 4:2:2 and alpha channel into 4;2:2:4 sampled video.

   Careful attention must be paid to disk bandwidth. The clips must be located on a filesystem that supports a bandwidth of about 22 MB/sec for uncompressed 4;2:2.

The 4:2:2 and alpha are generally not compressed so that they can be edited. The 4:2:2:4 video is generally compressed. Before recording any of these videos, you must set appropriate VST controls, as described in “Setting VST Controls” on page 207.

A code example showing how to play and record 4:2:2, alpha, and 4:2:2:4 is given in “Sample MVCP Scripts” on page 210.

Sample MVCP Scripts

The following sample code explains how to perform the following tasks:

- “Recording 4:2:2” on page 210
- “Recording the Alpha Channel” on page 212
- “Playing Alpha and Recording 4:2:2:4” on page 214

Recording 4:2:2

Use the following script to record 4:2:2:

```
send "UADD DIVO_DVC_2 * SHAR"
expect {
   timeout exit
   "202 OK"
}
expect {
   timeout exit
   -re "U\[0-9\]+"
}
set U1 $expect_out(0,string)

send "SET $U1 MED vtr.media.video.input.colorspace ccir601"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA SET $U1 MED vtr.media.video.input.compression.type none"
expect {
   timeout exitf
   "200 OK"
}

send "/SEQA SET $U1 MED vtr.media.clip.format default"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA SET $U1 MED vtr.media.video.input.format sdic”
expect {
   timeout exit
   "200 OK"
}

send "/SEQA SET $U1 MED vtr.media.video.input.packing R242_8f"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA LOAD $U1 b/$CLIP.4220.raw IN CRTE"
expect {
   timeout exit
   "202 OK"
}
expect {
   timeout exit
   "$CLIP"
}

send "/SEQA CUER $U1 0 +$SECS:00"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA REC $U1"
expect {
   timeout exit
   "200 OK"
}

send_user "\nType return to record the alpha component:\n"
interpreter
```

Recording the Alpha Channel

Use the following script to record the alpha channel:

```
# Set up record of alpha component

send "/SEQA SET $U1 MED vtr.media.video.input.packing 4_8"

expect {
   timeout exit
   "200 OK"
}

send "/SEQA LOAD $U1 $CLIP.0004.raw IN CRTE"
expect {
   timeout exit
   "202 OK"
}
expect {
   timeout exit
   "$CLIP"
}

send "/SEQA CUER $U1 0 +$SECS:00"
expect {
   timeout exit
   "200 OK"
}

#----------------------------------------
# Set up play of 422 component

send "UADD DIVO_DVC_0 * SHAR"
expect {
   timeout exit
   "202 OK"
}
expect {
   timeout exit
   -re "U\[0-9\]+"
}
set U2 $expect_out(0,string)

send "/SEQA SET $U2 MED vtr.media.video.output.format sdi"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA SET $U2 MED vtr.edit.preroll 5:00"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA SET $U2 MED vtr.edit.postroll 3:00"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA LOAD $U2 b/$CLIP.4220.raw"
expect {
   timeout exit
   "202 OK"
}
expect {
   timeout exit
   "$CLIP"
}

#----------------------------------------
# Go

send "/SEQA REVU $U2 0 +$SECS:00"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA UUWT $U1 $U2 SYNR"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA REC $U1"
expect {
   timeout exit
   "200 OK"
}
```

```
send_user "\nType return to setup the 4224 play and record:\n"
interpreter
```

Playing Alpha and Recording 4:2:2:4

Use the following script to play the alpha channel and record 4;2:2:4 on the same DIVO-DVC board:

```
# Set up play of alpha component

send "UADD DIVO_DVC_2 *"
expect {
   timeout exit
   "202 OK"
}
expect {
   timeout exit
   -re "U\[0-9\]+"
}
set U3 $expect_out(0,string)

send "/SEQA SET $U3 MED vtr.media.video.output.format sdi"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA LOAD $U3 $CLIP.0004.raw"
expect {
   timeout exit
   "202 OK"
}
expect {
   timeout exit
   "$CLIP"
}

send "/SEQA CUE $U3 0 +$SECS:00"
expect {
   timeout exit
   "200 OK"
}
```

```
#---------------------------------------
# Set up the record

send "/SEQA SET $U1 MED vtr.media.video.input.colorspace ccir601"

expect {
   timeout exit
   "200 OK"
}

send "/SEQA SET $U1 MED vtr.media.clip.format default"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA SET $U1 MED vtr.media.video.input.format sdi-dual-link"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA SET $U1 MED vtr.media.video.input.packing R2424_8"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA SET $U1 MED vtr.media.video.input.compression.sampling
4224"
expect {
   timeout exit
   "200 OK"
}

expect {
   timeout exit
   "$CLIP"
}

send "/SEQA CUER $U1 0 +$SECS:00"
expect {
   timeout exit
   "200 OK"
}
```

```
send_user "\nType return to do the 4224 play and record:\n"
interpreter
# Go

send "/SEQA REVU $U2 0 +$SECS:00"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA UUWT $U3 $U2 SYNR"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA UUWT $U1 $U2 SYNR"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA PLAY $U3"
expect {
   timeout exit
   "200 OK"
}

send "/SEQA REC $U1"
expect {
   timeout exit
   "200 OK"
}

send_user "\nType return to exit:\n"
interpreter
```

Configuring a Server for Clip Mirroring

You can set up one or more servers for clip mirroring with clip caches that mirror the clip cache on a designated primary server. This chapter describes how you configure such clip mirroring

You can use the VST *`clipmirror`* subsystem to implement a fully redundant cluster of two or more VST servers, each with its own private clip storage.

One server is designated the primary server and handles normal clip playback and recording, as well as file transfers to and from other non-VST servers or workstations. The VST *`clipmirror`* subsystem on each redundant (clip-mirroring) server synchronizes the contents of the redundant server's clip cache with the contents of the primary server's clip cache. That is, the clip caches on the redundant servers are maintained as mirrors of the primary server's clip cache.

This section consists of the following subsections:

Using Clip-Mirroring Servers

You set the name of the primary server that a redundant server is configured to mirror in the file *`/usr/vtr/config/system-defaults/clipmirror`* by specifying the value of the control vtr.clipmirror.primary_server.hostname. For example:

```
vtr.clipmirror.primary_server.hostname newprimary
```

The application can dynamically control the clip-mirroring feature by setting this control with the MVCP *SSET* command. For example:

```
SSET clipmirror vtr.clipmirror.primary_server.hostname newprimary
```

During the initial synchronization phase that occurs when VST starts on a redundant server, the *clipmirror* subsystem on the redundant server opens an MVCP connection to the primary server. Using this connection, the redundant server retrieves the list of clips (including modification times) from the primary server. The redundant server compares the primary server's clips with the clips in the local clip cache, checking for clips that are missing, out of date, or extraneous.

- If the file on a redundant server is newer than the file on the primary server, the clip is assumed to be up to date.

  If the clip file on a redundant server is of the same size as the corresponding file on the primary server and also has a later modification time, then the clip is assumed to be up to date.
- If a clip exists on the primary server only, the clip is copied via FTP to the redundant server.
- If a file exists on the redundant server only, a message is added to the VST log file (default */var/adm/vtr/logs/vtrlog*), but no other action is taken. The clip can be removed from the redundant server manually or by an application.

When a clip is recorded on or transferred via a network to the primary server, it is automatically copied to the clip-mirroring server. The speed of the copy operation depends on the bit rate of the clip, the network media connecting the servers, and whether other clips are being copied or are waiting to be copied to the redundant server.

The maximum number of FTP transfers spawned is set by the control vtr.clipmirror.max_threads. The value of this control should be based on the bandwidth of the network connection; the default is 20. The use of this control and other clip mirror controls is explained in "Setting Up Primary and Redundant Servers" on page 219; clip mirror controls are summarized in "System Controls for Clip Mirroring" in Chapter 3.

Once the clip caches on the primary and redundant servers are synchronized, the redundant server monitors the primary server for changes to the contents of its clip cache.

- Clips added to the primary server's clip cache are copied via FTP to the redundant server. A clip is copied to the redundant server again if the clip media data is modified on the primary server.

- If a clip is removed, renamed, or has its protection changed, the redundant server performs the same operation on its copy of the clip.

If the primary server becomes unavailable, each redundant server tries to reconnect to the primary server at an interval determined by the vtr.clipmirror.reconnect_interval control.

In a dual-server environment, if the primary server becomes unavailable for an extended period of time, you can reconfigure the redundant server to be the primary server. When the original primary server is available again, you can reconfigure it as the new redundant server. All new clips on the new primary server are then automatically copied to the new redundant server.

Note: The VST clip-mirroring feature does not work correctly with clip formats that require index files (such as vframe and stream).

Setting Up Primary and Redundant Servers

To set up primary and redundant servers, follow these steps:

1. To make sure that the filesystem holding the clips has the same real-time extent on the primary and on each redundant server, as root enter

   ```
   # xfs_growfs -n /usr/vtr/clips
   ```

   Output should resemble the following, which has been slightly reformatted:

   ```
   meta-data=/usr/vtr/clips/   isize=256    agcount=8, agsize=8192 blks
   data     =              bsize=16384  blocks=65536, imaxpct=25
            =              sunit=0      swidth=0 blks, unwritten=1
   naming   =version 1     bsize=16384
   log      =internal      bsize=16384  blocks=1000
   realtime =external      extsz=2097152 blocks=4445586, rtextents=34731
   ```

   In the last line, `extsz` is the extent size of the filesystem.

2. Using the *versions* command, make sure that the following subsystems are installed on the redundant servers:

   - *vst_eoe.sw.clipmirror*
   - *vst_eoe.sw.tools*

- *vst_eoe.sw.fsmon*
- *vst_eoe.sw.ftpd*

3. On the primary server, make sure the filesystems *vst_eoe.sw.fsmon* and *vst_eoe.sw.ftpd* are installed.
4. On each redundant server (not the primary server), open the configuration file */usr/vtr/config/system-defaults/clipmirror*. This file contains clip-mirroring controls on separate lines, commented out with pound signs (for example, `#vtr.clipmirror.primary_server.hostname ""`). Delete the pound sign and change the default primary server hostname (the null string) to the hostname of the primary server. For example, for a primary server named vst-1:

   ```
   vtr.clipmirror.primary_server.hostname vst-1
   ```

   If you do not want to set the name of the primary server as the default, the application can set the value of this and all other clip mirror controls with the MVCP *SSET* command; for example:

   ```
   SSET clipmirror vtr.clipmirror.primary_server.hostname vst-2
   ```

5. If desired, include a line in the configuration file */usr/vtr/config/system-defaults/clipmirror* on each redundant server to determine how often (in seconds) the redundant server tries to reconnect to a failed primary server. The minimum interval is 1 second; the default is 30. This example sets the interval to 60 seconds:

   ```
   vtr.clipmirror.reconnect_interval 60
   ```

6. If desired, set the value for the control vtr.clipmirror.max_threads on the redundant server(s); this control determines the maximum number of concurrent clip transfers from the primary server to the redundant server. The default limit is 20; the range is 1 to 100. For example:

   ```
   vtr.clipmirror.max_threads 2
   ```

7. On the primary and redundant servers, create a new user account named *vtrsync*:

   /usr/sysadm/privbin/addUserAccount -l vtrsync -u *idnumber* **-g 0 -H /usr/vtr/clips -S /bin/csh -P**

   The user ID (the variable *idnumber* in the example above) must be unique in the system; it can differ for different servers. To check user IDs already in the system, open */etc/passwd* and view the third field. In the following example, the user ID is 994.

   ```
   tutor::994:997:Tutorial User:/usr/tutor:/bin/csh
   ```

8. For the new user account vtrsync (on each server), set the password to vtrsync:

   ```
   # passwd vtrsync
   New Password:
   Re-enter new password:
   ```

   At each prompt shown above, enter **`vtrsync`**.

Stopping Automatic Backup

To stop the automatic backup, set the name of the primary server to an empty string via an MVCP connection to the redundant server(s):

```
SSET clipmirror vtr.clipmirror.primary_server.hostname ""
```

Note: Clips are always transferred in full between the primary and the redundant servers, regardless of any in and out points that have been set.

Redesignating Servers

If you are redesignating a redundant server as the primary server, follow these steps:

1. Turn off clip mirroring in the redundant server by unsetting the name of the primary server with the following command in an MVCP connection:

   ```
   SSET clipmirror vtr.clipmirror.primary_server.hostname ""
   ```

2. On the server that is to be the new primary server, open the configuration file *`/usr/vtr/config/system-defaults/clipmirror`* and make sure that the name of the primary is unset:

   ```
   vtr.clipmirror.primary_server.hostname ""
   ```

   Unset the primary if necessary.

3. On each redundant server, set the value of vtr.clipmirror.primary_server.hostname to the name of the new primary server.

An application can use the MVCP *`SSET`* command to change the values of the clip-mirroring controls while VST is running. For example, if vst-1 is the current primary server and vst-2 is the current redundant server, the following MVCP command swaps their roles:

- On vst-2, use

  ```
  SSET clipmirror vtr.clipmirror.primary_server.hostname ""
  ```

- On vst-1, use

  ```
  SSET clipmirror vtr.clipmirror.primary_server.hostname vst-2
  ```

The new redundant server, vst-1, starts the initial synchronization phase as described in “Using Clip-Mirroring Servers” on page 217.

Setting Transfers to Use Nondefault Network Interfaces

By default, spawned FTPs transfer clips between the primary interfaces of the primary and redundant servers. To enable these transfers to utilize another interface that might be higher bandwidth, use the controls vtr.clipmirror.local_server.hostname and vtr.clipmirror.primary_server.hostname in conjunction. FTP uses the server names specified in these controls.

For example, if the redundant server has a 100-Base-T Ethernet interface, vst1-enet, and a fibre channel connection, vst1-fc, you might wish to transfer clips on the higher bandwidth fibre channel connection. To do so, set vtr.clipmirror.local_server.hostname to vst12-fc. Similarly, if you want to use an interface on the primary server called p-vst-fc, set the control vtr.clipmirror.local_server.hostname to that name.

Archiving Clips

Asset management involves archiving, retrieving, deleting, and accessing media clips and the associated meta data that describes them. Because the SGI Media Server cache is limited in size, saving media files out to and retrieving files from an archival server is a common practice.

An archive system is a storehouse for information. It often serves multiple applications, such as asset creation, broadcasting, and compositing applications, as shown in Figure G-1. The media server is designed to work with one or more archive systems of the same type.

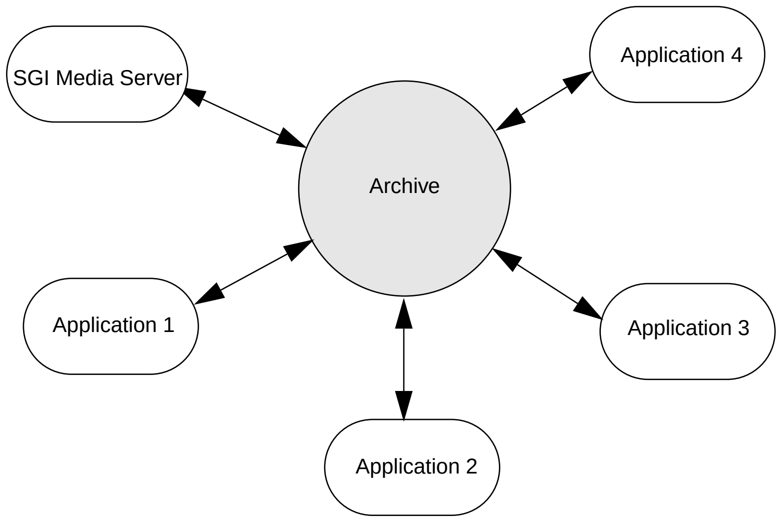


Figure G-1 Archive System

The SGI Media Server works in conjunction with StudioCentral 2.0 (SC 2.0) to archive media and associated meta data. The media server provides a set of asset management commands to interact with the archive.

For a more complete set of asset management tools, use the tools provided by the archive system. For more information about StudioCentral asset management tools, see the StudioCentral documentation.

This following sections describe the configuration of the archive system and the asset management tools you use with the media server to interact with that system:

- “Overview of the Archival System” on page 224
- “About StudioCentral” on page 226
- “Archive Interface Module” on page 227
- “Using the Media Server With the StudioCentral 2.0 Archive System” on page 227

Overview of the Archival System

The media server archives two entities:

- Media clips
- Meta data, which contains descriptive information about the media clips

Meta data contains information descriptive of the media clips associated with them. Information about the media clips includes the following items:

- Name
- Format
- Size
- Start time
- Duration

You use asset management tools to store and retrieve meta data. You might like to know, for example, the run time of a specific media file, or you might like to list the names of the media clips in the archive.

Configuration Overview

The elements in the archive system are the following:

- SGI Media Server
- Archive system
- Content server

All of these services can run on the same machine; however, it is recommended to keep the archive on separate, networked machines.

Chain of Communication

The following is the chain of communication when retrieving clips from the archive:

1. The media server asks an archive system to access a media asset.
2. The archive system contacts the correct content server.
3. If the content server is different from the SGI Media Server, the media is copied to the media server.

   If the content server is running on the media server, a link to the requested media is created.

Figure G-2 shows this communication path.

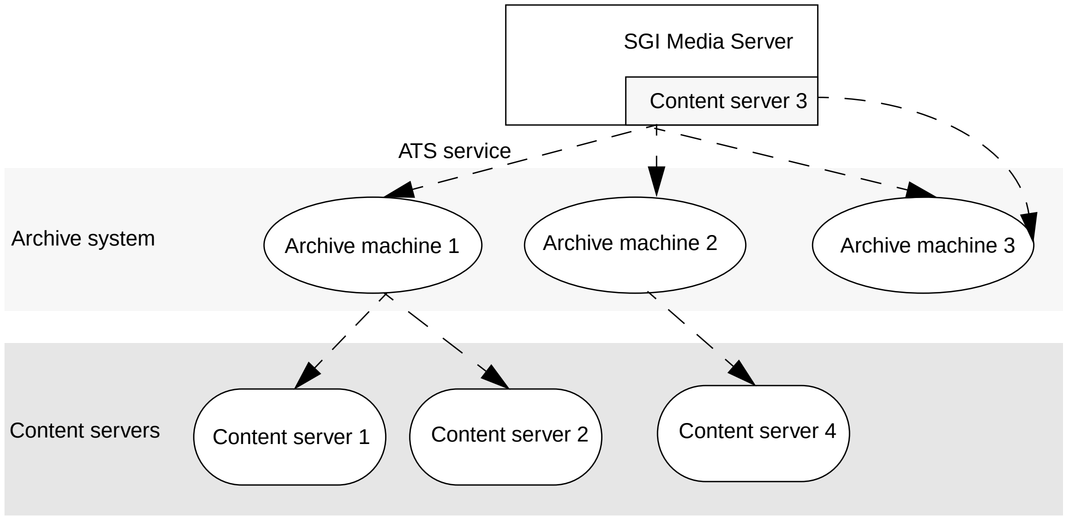


Figure G-2 Archive Configuration

Figure G-2 shows that the SGI Media Server can work with multiple archives. The servers that contain media content are usually not on the SGI Media Server; but not necessarily, as shown by Archive Server 3.

About StudioCentral

The media server can work with many archives. The archive system developed by SGI and supported by the SGI Media Server is StudioCentral, version 2.0 or later.

StudioCentral Digital Asset Management System (StudioCentral) is a library of C++ foundation classes that enables you to build multimedia-based applications with digital asset management capabilities.

For more information about StudioCentral, see the *StudioCentral Developer's Guide* (part number 007-3246-*nnn*).

Archive Interface Module

Video Server Technology (VST) has an archive interface that enables the media server to retrieve *clip*s from and store them in an archive asset repository. VST transfers the media to and from an archive system using the Asset Transfer Service (or ATS), communicating with ATS using the FTP-like ATS protocol.

Archive interface modules contain the code that is specific to a StudioCentral 2.0 archive system. These modules provide the support that is needed to locate a given *clip* in the StudioCentral 2.0 archive system and bring it into VST, and to store a clip from VST in the StudioCentral 2.0 archive system.

Using the Media Server With the StudioCentral 2.0 Archive System

Before the media server can successfully perform any archiving operation, the following guidelines must be met:

- The Informix or Oracle database server should be installed.
- The proper StudioCentral 2.0 images should be installed.
- All the CAS datamodels should be installed.
- The Informix or Oracle database server should be up and running.

  Ideally the database server should automatically start up at boot time.
- StudioCentral should be properly configured.
- At least one StudioCentral content server should be up and running.

  Ideally the content server should automatically start up at boot time.
- *`inetd`* should be configured to accept connection on behalf of the StudioCentral ATS service.

Archiving Tasks

When the media server is configured to use a StudioCentral 2.0 archive system, you can use the GUI mcclips to do the following:

- Locate *clip*s. See “Locating a Clip in the StudioCentral 2.0 Archive System” on page 229

- Bring a clip into the media server from the archive. See “Bringing In a New Clip From the Archive System” on page 230
- Store a clip in the archive. See “Using the Put Clip to Archive Window” on page 231

The remainder of this section shows how to use the GUI mcpanel to accomplish these tasks. However, you can also use the MVCP commands, shown in Table G-1 to perform these tasks and others.

Table G-1 Archive-Related MVCP Commands

Command	Description
AFND	Locate a clip by name in an available StudioCentral 2.0 archive system.
AFNG	Locate a pending or in-progress get-from-archive command for a clip.
AGET	Retrieve a clip from a StudioCentral 2.0 archive system.
ALSG	List the get-from-archive operations that are waiting or being processed.
ALSP	List the put-to-archive operations that are waiting or being processed.
APUT	Saves a clip to the archive system.

To learn more about using MVCP commands, see Appendix H.

Note: Only version 2.0 (or greater) of StudioCentral works with the SGI Media Server. Archival tasks require this application to be running.

Locating a Clip in the StudioCentral 2.0 Archive System

To display a list of media assets on StudioCentral, you must log into StudioCentral and use its tools.

To locate a clip in a StudioCentral 2.0 archive system, follow these steps:

1. Choose Archive > Find from the menu bar in mcpanel or from mcclips. The Find Clip in Archive window, as shown in Figure G-3, appears.

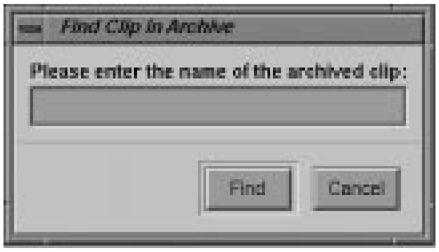


Figure G-3 Find Clip in Archive Window

2. Enter the clip's name, which corresponds to the value of the archived asset's **ClipId** attribute.
3. Click the *Find* button.

   To determine whether the clip was found, see if it appears in the media server log.

Bringing In a New Clip From the Archive System

To bring in a new clip from the archive system, follow these steps:

1. Choose Archive > Get New from the menu bar. The Get Clip from Archive window, as shown in Figure G-4, appears.

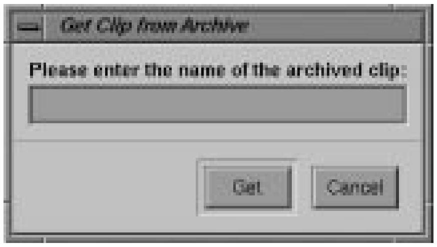


Figure G-4 Get Clip from Archive Window: Getting a New Clip

2. Enter the clip's name, which is the value of the archived asset's **ClipId** attribute.
3. Click the *Get* button.

A Get command can fail when there is no port available on the media server that supports the format of the clip.

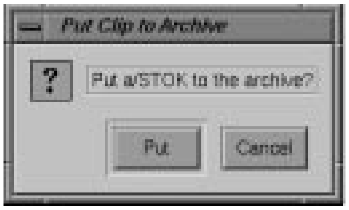
Note: If the clip exists in more than one archive system, the clip is brought in from the first archive system in which it is found. The clip can be brought in from a specified archive system using the MVCP command *AGET*.

Using the Put Clip to Archive Window

To write a clip to a the StudioCentral 2.0 archive system, follow these steps:

1. Do one of the following:
    - Select the clip in the Clip Manager window and then choose Archive > Put from the menu bar.
    - If the clip information window for the clip is displayed, click the *Put Archive* button in the information window.

   The Put Clip to Archive window, as shown in Figure G-5, appears.



Put a/STOK to the archive?

Figure G-5 Put Clip to Archive Window

2. Click the *Put* button to write the clip to the StudioCentral 2.0 archive system.

Note: The clip is written to the first StudioCentral 2.0 archive system. That is, if there is more than one StudioCentral 2.0 archive system, the clip is written to the first one. The clip can be put into an StudioCentral 2.0 archive system other than the first one by using the MVCP command *APUT*.

Checking Success of Storing Clip

Hardware, network, and StudioCentral failures can contribute to failures in storing clips in an archive. The failure. however, may not be reported. You might, for example, be able to AFND a clip on the archive because there is a “ghost” of the clip there, but not AGET it because the clip is really not there. If you find that you cannot AGET a clip, you know you need to use the Put Clip to Archive Window (or APUT) again to store the clip.

Multiport Video Computer Protocol (MVCP) Command Summary

The base Video Server Technology (VST) software includes a protocol processor, which provides full-featured control of VST through a textual TCP/IP-based protocol called the Multiport Video Computer Protocol, or MVCP.

This appendix describes the MVCP commands.

Protocol Format

MVCP is a simple request/response protocol which is implemented over a TCP byte-stream connection (i.e., a stream socket). The protocol is similar in nature to the FTP control protocol in that requests consist of two- to four-character commands with a varying number of space- delimited arguments terminated by a carriage-return/line-feed.

Responses consist of a textually-represented numeric result code with an associated informational message terminated by a CR/LF. Success is indicated with a response code of the form 2xx. Unless otherwise stated, a command will return the result code 200 if the command is successful.

Unlike FTP, certain MVCP responses may also include a single response line or multiple response lines terminated by a single blank (CR/LF- only) line. The successful result code is 202 if a single response line is returned and 201 if a response list (one or more lines terminated by a blank line) is returned.

All MVCP command arguments and responses are space-delimited. Command or response arguments which contain spaces are surrounded by double quotation marks. In some cases, response arguments that do not contain spaces may be double-quoted, so the application should always be prepared to handle them.

Note: Future enhancements to MVCP may result in additional arguments being added to commands or responses. 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.

An application should not depend on the exact number of arguments on a response line returned by an MVCP command as additional arguments may be added as future enhancements to the protocol. An application should expect and process those arguments which it understands but should be prepared to ignore additional arguments which it does not expect and understand.

Unit Management

A single MVCP connection can control multiple VST units (each unit is a logical VTR transport capable of loading, cueing, and playing a clip). MVCP commands which pertain to a specific unit include the unit name as the first argument of the request while global commands, commands which do not pertain to a specific unit, do not include a unit name.

For example:

```
CINF clip1 (global: retrieve clip information)
LOAD U1 CLIP3.DIF OUT (unit: load clip into unit U1)
```

When the unit name is omitted or specified as '*' with a unit command, the unit most recently created 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 controller (which could be another MVCP control connection or another external control processor such as a station automation system or on-line editor). Of course, due care must be exercised when sharing control of a unit between two controllers. Interfering with a unit owned by one of the VST external protocol processors (for example, Sony or Louth) will lead to unpredictable behavior.

Establishing an Interactive MVCP Connection

To establish an interactive MVCP connection, use telnet to establish a control connection to the MVCP port on the media server.

The following example establishes an interactive connection to the MVCP port (which is usually port 5250) on the media server “mediaserver.” The MVCP *PLS* (List Ports) command is run and then the *BYE* command is used to terminate the control connection:

```
% telnet mediaserver 5250
Trying 133.144.166.34...
Connected to mediaserver.abc.com.
Escape character is '^]'.
100 VTR Ready
PLS
201 OK
DIVO_DVC_0 BOTH "SGI XT-DIVO Digital Video Option" VID
DIVO_DVC_1 BOTH "SGI XT-DIVO Digital Video Option" VID
...
BYE
200 Goodbye
Connection closed by foreign host.
%
```

Rules for Using MVCP Commands

The following rules apply to the use of the MVCP commands:

- The commands are case-sensitive and must be entered exactly as shown in this chapter.
- All MVCP command arguments are space-delimited. Command arguments that contain spaces should be surrounded by double quotation marks.
- Each command must end with a carriage return and line feed.
- The command set is divided into two subsets. One subset contains the global commands, which do not pertain to a specific *unit*. The other subset contains the unit commands, which pertain to a specific unit. (See “Global Commands” on page 242 and “Unit Commands” on page 263 for detailed information about these command subsets.)
- MVCP commands that pertain to a specific *unit* contain the unit name as the first argument of the request. Global commands do not include a unit name. For

example, the following is a global command that requests information about the *clip* named "NA40125D." This command does not have a unit name:

```
CINF NA40125D           (get clip information)
```

The following is a unit command that loads the clip named "NA40125D" for output into the unit identified by "U1":

```
LOAD U1 NA40125D OUT           (load clip into unit U1)
```

- MVCP responds to each command with a response header line, which is terminated with a carriage return and line feed. All MVCP responses are space-delimited. Response arguments that contain spaces are surrounded by double quotation marks. (See "Response Codes" on page 238 for information.)

Note: Future enhancements to MVCP may result in additional arguments being added to commands or responses. The additional arguments will be added in such a way that the commands or responses will be backward-compatible with the existing commands or responses, so that existing applications will continue to work correctly.

Command Sets

The protocol command set is divided into two subsets. One subset includes the global commands, those commands which do not pertain to a specific unit. The other subset includes the unit commands.

In general, global commands affect the global state of VST and thus all control connections (through MVCP or other protocols). However, certain commands affect only the connection on which the commands are executed.

Displaying MVCP Commands

One way to learn MVCP commands is to see how they are used in the applications included in VST, such as mcstat, mcpanel, and mcclips. To display in the console window the MVCP commands as they are sent to VST and the MVCP responses as they are received from VST, start the applications with the **-v3** option:

```
# mcpanel -v3
```

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.

Note: On video devices, command timing is ignored when the unit is not in play or record mode.

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 defaults to 29.97 and can be set with FRAT). For instance, 14:14:00:00 specifies that the command should start at exactly 2:14pm local time.

Optionally, the timecode may also include a date by prepending the timecode with yyyymmddT (for example, 19980828T12:34:00:02).

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: f the system is receiving house timecode, 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. Some commands also return either a single response data line or multiple response data lines. When multiple data lines are returned, they are terminated by a single blank line.

The response header line has the following format:

xxx   *message_text*

where

- *xxx* is a three-digit decimal response code.
- *message_text* is a textual message describing the response.

This response header line may be followed by one or more data lines, depending on the response code.

Each of the following is an example of a response header line:

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

The following is an example of a response header line that is followed by two data lines: In this example, a blank line follows the second data line.

```
ULS
201 OK
U1 mvcp/originserver mvp BOTH * DONE IDLE * 0 *
U2 mvcp/originserver mvp BOTH * DONE IDLE * 0 *
```

Caution: Additional fields may be added as future enhancements to response lines. The additional fields will be added in such a way that the responses will be backward-compatible so that existing applications will continue to work correctly.

The response codes are divided into the following categories:

- 2xx, successful command execution
- 4xx, command format or setup errors
- 5xx, command execution errors

Some error codes have more than one error message. The message depends on the error.

Response Text

```
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

400 Command invalid
401 Port name missing
402 Unit name missing
403 Clip not found
403 Unit not found
403 Unit not open
404 Access mode missing or invalid
405 Clip name missing
405 New clip name missing
406 Command not supported
407 Load mode missing or invalid

410 Invalid clip time
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 play speed
418 Invalid speed/count
419 Invalid number of passes

420 Must specify start time
421 Bad create flag
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
434 Unable to add clip
435 Filename missing
436 f1f2 has to be in CAP
436 Still-frame mode invalid
436 Frame interleave mode invalid
437 Cannot specify both in and out as relative
438 Unit already open
439 Sync modifier not supported for this command

440 Invalid frame-rate
448 Too many arguments
457 User name missing
458 Password missing
460 Must provide USERname first

500 General error (message varies)
501 Unable to copy clip
501 Unable to find clip in archive
501 Unable to find pending get-from-archive request
501 Unable to issue delete-from--archive request
501 Unable to issue get-from-archive request
501 Unable to issue put-to-archive request
501 Unable to link clip
501 Unable to rename clip

510 General error (message varies)
510 Invalid clip time
511 General error (message varies)
540 Invalid frame-rate
548 Too many arguments
```

OK Response

After issuing a command, the system may respond, “OK.” This response means that the command has been parsed, not that it has been executed. For example, if you:

```
LOAD Unit clipName
```

without a mode or CRTE argument, and clip does not exist, the command returns an “OK,” even though a CRTE is required.

Whether or not the command will succeed depends on things such as:

- What type of media port is being controlled.
- Whether or what type of clip is loaded.
- What state the unit is in.
- What state the hardware is in.

It is impossible for MVCP to know anything beyond the syntactic validity of a unit command.

One way of getting a response when the command has been executed is to issue the command with /SYNC. For example, if you:

```
/SYNC LOAD Unit clipName
```

an error message is returned immediately.

Using P2 Commands with MVCP Commands

You cannot mix MVCP and P2 commands. You must either control a port using only MVCP commands (LOAD/CUER/REC), or the P2 protocol (AUTOEDIT or CUE/PLAY/EDIT_ON/EDIT_OFF).

There is one exception: you can use the following MVCP command to load a new clip onto a port being controlled through P2:

```
SSET <control-port> vtr.control.clip.name <clip-name>
```

where *control-port* can be vtr_1, vtr_2, etc.

Global Commands

Global commands affect the entire system, as opposed to unit commands described in "Unit Commands" on page 263 that only effect specified units.

There are many different kinds of global commands:

- "Access Control" on page 244
- "Ports" on page 245
- "Units" on page 246
- "Archive Management" on page 249
- "Clip Management" on page 253
- "System Controls" on page 259
- "Statistics" on page 260
- "Miscellaneous" on page 261

Table H-1 identifies the global MVCP commands that are documented in this section.

Table H-1 Global MVCP Commands

Command	Function
AFND	Locate a clip in an available a StudioCentral 2.0 archive system.
AFNG	Locate a pending or in-progress get-from-archive command for a clip.
AGET	Retrieve a clip from a StudioCentral 2.0 archive system.
ALSG	List the get-from-archive operations that are waiting or being processed.
ALSP	List the put-to-archive operations that are waiting or being processed.
APUT	Store a clip to a StudioCentral 2.0 archive system.
ARM	Delete a clip from a StudioCentral 2.0 archive system.
ARMG	Cancel all pending or in-progress get-from-archive operations.
BYE	Terminate the current control connection.
CADD	Add a clip that has been inserted into the clip cache through an external means.

Table H-1 (continued) Global MVCP Commands

Command	Function
CCHP	Add specified protections to, or remove them from, a clip.
CCP	Create a new clip by copying it from an existing clip.
CCST	Get the current status of the VST Clip Cache.
CEDP	Set the edit points for a clip.
CGP	Get clip protections.
CIMG	Copy an image to a file.
CINF	Get information about a clip.
CLN	Create a new clip that shares the clip media content of another clip.
CLS	Get a list of all the clips in the VST clip cache.
CLSA	List the clips that have been added to the clip cache since the last time the CLSA command was issued or since the CMON command that created this clip monitor was executed.
CLSR	List the clips that have been removed from the clip cache since the last time the CLSR command was issued or since the CMON command that created this clip monitor was executed.
CMIN	Get the number of clips that have been added to, or removed from, the clip cache.
CMON	Initiate monitoring of the clip cache.
CMV	Rename a clip.
CRM	Delete a clip from the clip cache.
CRMA	Delete all clips currently residing in the clip cache.
ERR	Get the code and description of the last global error that occurred for a controller.
FRAT	Set the frame rate used in translating timecodes for command timing.
GTOD	Get the current system time.
MON	Place control connections into event monitoring mode.
PASS	Sets the password for access control.
PLS	Get the list of supported media ports.

Table H-1 (continued) Global MVCP Commands

Command	Function
SGET	Retrieves the values of system controls for the subsystem(s).
SORD	Set the sort order that's used when lists are returned.
SSET	Sets system controls for the subsystem.
STLS	List statistics.
STOD	Set the system time.
STST	Calculate various statistics.
STZ	Reset statistics.
UADD	Create a new unit.
ULS	Get a list of the existing VST units.
USER	Sets the user for access control.

Access Control

This section lists those global commands that deal with system access.

USER *username*

The USER command sets the *username* for access control purposes.

The USER command applies only to the current MVCP connection.

PASS *password*

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

The PASS command applies only to the current MVCP connection.

Ports

This section lists those global commands that deal with ports.

PLS

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

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

Where:

- *name* is the name of the port.
- *mode* is the port's input/output mode (IN, OUT, or BOTH).
- *description* is a description of the port.
- *type* is the type of the port, one of NET (network), VID (video), DECK (deck control), or DISK (disk storage).
- *port-physical-name* is the physical name of the media port.

For example:

```
PLS
201 OK
DIVO_DVC_0 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_0
DIVO_DVC_1 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_1
DIVO_DVC_2 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_2
DIVO_DVC_3 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_3
DIVO_DVC_4 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_4
DIVO_DVC_5 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_5
DIVO_DVC_6 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_6
DIVO_DVC_7 BOTH "SGI Digital Video Option (DVC)" VID DIVO_DVC_7
```

Units

This section lists those global commands that affect all units in the system.

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

The UADD (Add Unit) command creates a new unit. 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 must be opened without a storage port (for example, deck-control ports). 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) means only one unit may exist which uses the media port at any given time. If a unit already exists for that port, the UADD will fail.

Shared access (SHAR) means multiple units may be created which all access 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) means multiple units may use the media port at the same time. This mode is used most commonly on multiplexed networking ports (such as ATM).

The *owner-info* is optional and sets the owner info for the new unit. The owner info is included in certain status commands such as UINF, UGIN, and ULS.

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

Note: An application must not make any assumptions about the name of the unit including the format of the name. Unit naming is subject to change in future releases.

ULS

The ULS (List Units) command returns a list of the existing VST units. A snapshot of the state of each unit is included.

If the ULS command is successful, the response code is 201 and one response line for each unit is returned followed by a blank line. The following is the format of each unit state:

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

Where:

- *name* is the name of the unit.
- *owner* is the name of the controller that created the unit.
- *port* is the name of the unit's media port.
- *mode* is the mode of the media port (IN, OUT, or BOTH).
- *clip* is the name of the loaded clip ("*" if no clip is loaded).
- *status* is the status of the current function (BUSY is the initial state, RUN is the running state, DONE is the completed state, ERR is the error state).
- *function* is 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), REVU (Review), EDIT (Edit), RHRS (Rehearse).
- *location* is the current clip location in the format hours:minutes:seconds:frames. In drop-frame mode, the final colon (:) is replaced by a period (.).
- *speed* is the current playback speed (1000 = normal 1x speed).
- *frame-rate* is the frame-rate (in frames/sec) of the current clip. 525-line timing is specified in its approximate form, 29.97.
- *command-id* is the id of the command currently being processed by the unit.

For example:

```
ULS
201 OK
U1 mvcp/originserver mvp BOTH * DONE IDLE * 0 *
U2 mvcp/originserver mvp BOTH * DONE IDLE * 0 *
```

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

The MON (Monitor) command places the current control connection 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, unit events for all 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 removed), UCHG (unit state change), ULOC (unit location change), UCTL (unit control change), UERR (unit error), CADD (clip added), CRM (clip removed). Only the specified types of events are returned.

If no event-type list is specified, UCHG, URM, UERR, and UCTL events are returned, and if no units are specified, UADD events are also returned.

Event monitoring is terminated by closing the control connection.

The format of the event response line is one of the following:

UADD *owner* *port* *mode* *port-physical-name*

- *owner* is the name of the controller that created the unit.
- *port* is the name of the unit's media port.
- *mode* is the mode of the media port (IN, OUT, or BOTH).
- *port-physical-name* is the physical name of the media port controlled by the unit.

CADD *clip* *format* *size* *resident-size* *start* *end* *in* *out* *frame-rate*

- *clip* is the name of the clip.
- *format* is the format of the clip.
- *size* is the size of the clip content in bytes.
- *resident-size* is the size of the content that is currently resident on the clip cache disk system.
- *start* is the timecode of the first frame of the clip.
- *end* is the timecode of the frame after the last frame of the clip.
- *in* is the current mark-in point of the clip.

- *out* is the current mark-out point of the clip.
- *frame-rate* is the frame-rate (in frames / sec) of the current clip.
- 525-line timing is specified in its approximate form, 29.97.

UOPN *unit-name*

The UOPN (Unit Open) command opens an existing unit and makes it controllable by this MVCP connection.

If the unit is opened successfully, the response code is 202 and a single response line is returned containing the name of the opened unit.

Warning: Opening a unit owned by a Sony or Louth port using MVCP commands (for example, UOPN) causes unpredictable behavior.

Archive Management

This section lists those global commands that deal with archiving clips.

AFND *clip*

The AFND (Find Clip in Archive) command attempts to locate the clip specified by *clip* in an available archive system. If successful, the response code is 201, and a response line is returned for each archive system in which the clip is found. The response format is:

```
*archive-type* *archive-host*
```

Where:

- *archive-type* is the type of archive system.
- *archive-host* is the specific host name of the archive system containing the specified clip.

For example:

```
AFND NA1001
201 OK
mediahub mhhost
```

AFNG *clip*

The AFNG (Find Archive Get) command attempts to locate a pending or in-progress archive get command for the clip specified by *clip*. If found, the response code is 202, and a single response line is returned in the following format:

```
*clip* *atype* *ahost* *requester* *requested* *started* *eta*
```

Where:

- *clip* is the name of the clip.
- *atype* is the type of archive system.
- *ahost* is the specific host name of the archive system.
- *requester* is the name of the controller requesting the clip.
- *requested* is the time the retrieval was requested.
- *started* is the time the retrieval was started.
- *eta* is the time the retrieval is expected to complete.

AGET *clip* [*archive-clip-name* *archive-host*]

The AGET (Get from Archive) command enqueues a get-from-archive operation to retrieve the clip specified by *clip* from an attached archive system. If *archive-clip-name* is not specified or the default *archive-clip-name*, '*', is specified, *clip* will be used.

If *archive-host* is specified, VST attempts to retrieve the clip from only that specific archive. Otherwise, the clip is retrieved from the first archive in which it can be located.

If a get-from-archive operation is already pending for the clip, AGET does not add a new one.

AGET always returns immediately with response code 200.

Archive operations are processed in-order by the VST Archive Manager which can process a configurable number of concurrent operations for each attached archive system.

If the clip cannot be successfully retrieved from the archive, the failure will be logged in the VST status log. In addition, a clip monitor or event monitor will see the failing clip as having been removed from the clip cache.

ALSG The ALSG (List Archive Gets) command lists the get-from-archive operations that are waiting or being processed. The response code is 201, and a single response line is returned for each archive request in the following format:

```
*clip* *atype* *ahost* *requester* *requested* *started* *eta*
```

Where:

- *clip* is the name of the clip.
- *atype* is the type of archive system.
- *ahost* is the specific host name of the archive system.
- *requester* is the name of the controller requesting the clip.
- *requested* is the time the archive operation was requested.
- *started* is the time the archive operation was started.
- *eta* is the time the archive operation is expected to complete.

ALSP

The ALSP (List Archive Puts) command lists the put-to-archive operations that are waiting or being processed. The response code is 201, and a single response line is returned for each archive request in the following format:

```
*clip* *atype* *ahost* *requester* *requested* *started* *eta*
```

Where:

- *clip* is the name of the clip.
- *atype* is the type of archive system.
- *ahost* is the specific host name of the archive system.
- *requester* is the name of the controller requesting the clip.
- *requested* is the time the archive operation was requested.
- *started* is the time the archive operation was started.
- *eta* is the time the archive operation is expected to complete.

APUT *clip* [*archive-clip-name* [*archive-host*]]

The APUT (Put to Archive) command enqueues a put-to-archive operation to store the clip specified by *clip* to an attached archive system. If *archive-clip-name* is not specified or the default *archive-clip-name*, '*', is specified, *clip* will be used.

If *archive-host* is specified, the clip is stored into that specific archive. Otherwise, the clip is stored into the default archive.

If a put-to-archive operation is already pending for the clip, APUT does not add a new one.

If the specified clip does not exist, APUT returns an appropriate error response code. Otherwise, APUT immediately returns response code 200.

Archive operations are processed in-order by the VST Archive Manager which can process a configurable number of concurrent operations for each attached archive system.

If the clip cannot be successfully stored into the archive, the failure will be logged in the VST status log.

ARM *archive-clip-name* [*archive-host*]

The ARM (Delete from Archive) command enqueues a delete-from-archive operation to delete the clip specified by *archive-clip-name* from an attached archive system. If a delete-from-archive operation is already pending for the clip, ARM does not add a new one.

ARM always immediately returns response code 200.

Archive operations are processed in-order by the VST Archive Manager which can process a configurable number of concurrent operations for each attached archive system.

ARMG

The ARMG (Cancel Archive Gets) cancels all pending or in-progress get-from-archive operations.

ARMG always immediately returns response code 200.

Clip Management

This section lists those global commands that deal with clip management.

CADD *clip* [*format*]

The CADD (Add Clip) command adds a new clip that has been inserted into the clip cache through an external means. The name of the new clip is specified by *clip*. The format of the clip is specified by *format*. If *format* is not specified, the clip file's *clip.format* attribute will be used, if it exists, or VST will attempt to automatically detect the format of the clip.

The clip media file must be placed in the appropriate location in the clip cache before issuing the CADD command. If the clip format requires an index, the index file must also be placed in the appropriate index directory before issuing the command.

Note: CADD is no longer required to make a clip available to other VST commands (such as loading it into a unit). VST will automatically attempt to locate and add a clip whenever a controller references the clip.

However, CADD may be used to add a clip to VST's internal clip tables such that it will appear in the clip listing returned by CLS. CADD will also generate the appropriate clip-add event messages for controllers that are monitoring the clip cache.

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

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

The types of protection include ATTR (Attribute Protect), MV (Rename Protect), REC (Record Protect), and RM (Delete Protect). 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 an new clip named *new-clip* by copying the clip specified by *clip*. After the copy is created, there is no association between the new clip and the original clip.

If successful, the response code is 200.

CCST

The CCST (Clip Cache Status) command returns the current status of the VST 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* is the number of clips resident in the clip cache.
- *bytes-used* is the number of bytes used on the storage systems which hold the clip cache.
- *bytes-avail* is the number of free bytes available on the storage systems which holds the clip cache.
- *bytes-avail-contiguous* is the number of free bytes available on the storage system which has the greatest amount of free space.

CEDP *clip* *mark-in* *mark-out*

The CEDP (Set Clip Edit Points) command sets the edit points for the clip specified by *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.

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

CGP *clip*

The CGP (Get Clip Protection) command returns the current protection of the clip specified by *clip*.

If successful, the response code is 202, and a single response line is returned which contains a list of the protections currently enabled for the clip. The protections are ATTR (Attribute Protect), MV (Rename Protect), REC (Record Protect), and RM (Delete Protect).

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

The CIMG (Create Clip Image) command extracts the image data associated with the frame specified by *timecode* of the clip specified by *clip* and writes the image to the file specified by *filename*.

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 "rgb", "yuv", and "tiff" (or others as supported by the Image Format Library). If *format* is not specified, the format is inferred from the filename extension (".rgb", ".yuv”, ".tiff").

For example:

```
CIMG bigclip 00:01:00:02 F1F2 x.dif dif
200 OK
```

CINF *clip*

The CINF (Clip Info) command returns the attributes of the clip specified by *clip*. If the clip currently exists in the clip cache, 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* *frame-rate*
- *clip* is the name of the clip.
- *format* is the format of the clip.
- *size* is the size of the clip content in bytes.
- *resident-size* is the size of the content that is currently resident on the clip cache disk system.
- *start* is the timecode of the first frame of the clip.
- *end* is the timecode of the frame after the last frame of the clip.
- *in* is the current mark-in point of the clip.
- *out* is the current mark-out point of the clip.
- *frame-rate* is the frame-rate (in frames / sec) of the clip. 525-line timing is specified in its approximate form, 29.97.

CLN *clip* *new-clip*

The CLN (Link Clip) command creates a new clip named *new-clip* which shares the clip media content of the clip named by *clip*. The clip attributes (such as edit points) of the new clip may be set independently of the original clip.

If the original clip is deleted, the new clip still retains the content of the original clip 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 The CLS (List Clips) command returns a list of all the clips in the VCP- Recorder clip cache. The response code is 201 and one response line is returned for each clip in the cache. The format of the response line is:

- *clip* *format* *size* *resident-size* *start* *end* *in* *out* *frame-rate*
- *clip* is the name of the clip.
- *format* is the format of the clip.
- *size* is the size of the clip content in bytes.
- *resident-size* is the size of the content that is currently resident on the clip cache disk system.
- *start* is the timecode of the first frame of the clip.
- *end* is the timecode of the frame after the last frame of the clip.
- *in* is the current mark-in point of the clip.
- *out* is the current mark-out point of the clip.
- *frame-rate* is the frame-rate (in frames / sec) of the clip. 525-line timing is specified in its approximate form, 29.97.

CLSA

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

Before the CLSA command may be issued by a controller, the CMON (Clip Monitor) command must be issued to initiate monitoring of the clip cache for this controller.

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.

CLSR

The CLSR (List Removed Clips) command lists the clips that have been 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).

Before the CLSR command may be issued by a controller, the CMON (Clip Monitor) command must be issued to initiate monitoring of the clip cache for this controller.

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 that have been added to and/or removed from the clip cache and will be 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* is the number of clips that have been added.
- *num-clips-removed* is the number of clips that have been removed.

CMON

The CMON (Clip Monitor) command initiates monitoring of the clip cache. The CLSA and CLSR commands are used to retrieve, respectively, the clips that have been added to 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 the clip specified by *clip* to a new name specified by *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 the clip specified by *clip*. If any units currently have the clip loaded, actual deletion of the clip will be deferred until all units have unloaded the clip.

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

CRMA

The CRMA (Delete All Clips) command deletes all clips currently residing in the clip cache. Clips which are currently being retrieved from an archive system may or may not be deleted depending on the progress of the retrieval.

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

System Controls

This section lists those global commands that deal with system controls.

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

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

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*"
```

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

Statistics

This section lists those global commands that deal with statistics gathering.

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* is the name of the instance of the component that is generating the statistic.
- *statistic name* is the name of the statistical value.
- *value* is the current value of the integer or floating-point statistic. The value of certain types of statistics (for example, 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* is the number of statistical values matching the pattern.
- *samples* is the total number of samples collected.
- *min* is the minimum value.
- *max* is the maximum value.
- *sum* is the sum of the values.
- *mean* is the mean of the values.
- *stddev* is 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.

Miscellaneous

This section lists all global commands not coverecd in the previous sections

BYE

The BYE command terminates the current control connection.

ERR

The ERR command returns the code and description for the last global error that occurred for this controller. 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* is the error code.
- *description* is the error description.

FRAT *frame-rate*

The <FRAT> (Frame Rate) command sets the frame rate used in translating timecodes for command timing. The *frame-rate* is specified as frames per second. Supported values are 24, 25, 29.97, and 30.

The FRAT command applies only to the current MVCP connection.

GTOD

The GTOD (Get Time-of-Day) command returns the current VST 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*

For example:

```
GTOD
202 OK
14:24:01.11 19980105T222400.890307Z 94038459071434
```

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 VST 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.

Table H-2 identifies the MVCP *unit* commands that are described in this section. .

Table H-2 MVCP Unit Commands

Command	Function
CUE	Cue for playback the clip currently loaded in a unit.
CUER	Cue for recording the clip currently loaded in a unit.
EDIT	Cue for recording and then begin recording.
FCLR	Erase the audio and video from specified frames in a clip.
FF	Fast forward the unit.
FINS	Move frames within a clip.
FNEW	Insert blank frames within a clip.
FOVR	Overwrite frames within a clip.
FRM	Remove frames from within a clip.
GET	Get the values of controls.
GOTO	Jump a unit's transport to a specified location.
JOG	Move a unit forward or reverse by the number of frames.
LIMS	Modify the edit limits being used by the current playback or record function.
LOAD	Load a clip into a unit.
PAUS	Pause a unit.
PLAY	Begin playback on a unit.
REC	Begin recording on a unit or resume normal recording after a PAUS command.
REVU	Do the equivalent of a cue for playback and then begin playing.
REW	Fast reverse the unit.

Table H-2 (continued) MVCP Unit Commands

Command	Function
RHRS	Cue for playback and then begin playing, with the media port switched to end-to-end mode for playback.
RSUM	Resume playback or recording.
SET	Set controls for the unit.
SHTL	Shuttle a unit at a specified speed.
STON	Sets the speed of the unit specified.
STON	Halts playback but does not decue the unit.
STOP	Stop playback or recording on a unit.
UCLS	Close an opened or created unit.
UERR	Get the code and description of the last error that occurred on a unit.
UFLS	Flush the command queue for a unit.
UGIN	Get the owner and port information for a unit.
UINF	Get the owner and port information for a unit.
UINT	Interrupt the command currently being processed by a unit.
ULOC	Return the current transport location.
UNLD	Unload the clip currently loaded in a unit.
UOPN	Open an existing unit and make it controllable.
USTA	Get the status of a unit.
USYN	Set the default synchronization mode for a unit specified.
UUWT	Synchronize command execution between two units.
UWAT	Wait for the completion of the last command that was issued to a unit.

Before you can use these *unit* commands, you must either create the unit or open it, according to the following:

- When you want to use a unit that does not already exist, you create the unit using the global *UADD* command.

  When you create the unit, you identify the media *port* and you specify the port access mode, which determines how access is shared among multiple units that access the same media port. You can specify exclusive access, shared access, or concurrent access of the media port.

  When you create the unit it is automatically opened.

- When you want to share a unit that has already been created, you open the unit using the global *UOPN* command.

  When you open a unit, it must have previously been created by another control port. Units can be shared with either another MVCP control connection or with another external control processor, such as a station automation system.

Caution: Extreme care must be exercised when sharing control of a unit between two controllers. Interfering with a unit owned by one of the automation protocol processors (for example, a Louth controller) leads to unpredictable behavior.

Unit Command Modes

The VST units execute asynchronously with the control processors that are controlling them. How the MVCP control processor interacts with the units it controls and the client application that is issuing the MVCP commands is determined by the 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 executing. Also by default, the MVCP control processor returns a response to the client as soon as the command has been queued for the unit, so the OK response to a command means only that the command was successfully queued, not that the command completed successfully or has even started executing.

Synchronization Mode

The synchronization mode determines when the MVCP control processor returns a command response to the MVCP client. The available synchronization modes are: ASYN (async), SYNI (sync-init), SYNR (sync-run), and SYNC (sync-complete). Async (ASYN) mode is as has been described: the control processor responds as soon as the command is queued.

Sync-init (SYNI) mode delays the response until the unit has reached the initialization state on the command. This does not mean that the unit has begun to process the command, and in fact it may have to wait for a shared resource (such as the media port) to become available before it can continue.

Sync-run (SYNR) mode delays the response until the unit has reached the "running" state on the command. This means that the unit is actively processing the command. For instance, the running state for a PLAY command means that video is actually being played.

Sync-complete (SYNC) mode delays the response until the unit has reached the "complete" state on the command. This means that the unit will do no further processing for this command. If the command is CUE, the complete state is reached when the unit has been fully cued and is ready to being playback. If the command is PLAY, the complete state is reached when playback has finished.

If no error occurs in the unit before the desired sync state has been reached, the response code is 200 (or 201, or 202, as appropriate). If an error occurs before the unit reaches the desired state, a 5XX error response code will be returned.

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 has completed executing.

Command Sequencing

Command sequencing refers to the way in which consecutive commands are issued to and processed by the target unit. There are two variables which determine how a command is issued and processed: the preemption mode and the queuing mode.

Preemption

The preemption mode determines whether a command will preempt the previous command or wait for the previous command to reach a certain status before being executed by the unit. Preemption is controlled by both the "previous" command and the "next" command.

The "previous" command may be issued with a qualifier that defers execution the "next" command until the "previous" command has reached a certain 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), NRUN (defer until previous Running), NCMP (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.

Unit Commands

This section lists the commands that affect specific units.

CUE [*unit-name* [*in* [*out* [*direction* [*passes*]]]]]

The CUE (Cue For Play) command cues for playback the clip currently loaded in the unit specified 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, 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" all refer to the same 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 stop).

If *passes* is specified as -1, the unit is cued in free-range mode (only on 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.

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

The CUER (Cue For Record) command cues for recording the clip currently loaded in the unit specified 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 be specified.

If either *in* or *out* is specified, 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" all refer to the same edit range.

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.

FCLR *unit-name* *in* *out*

The FCLR (Clear Frames) command clears (erases) the frames of the clip loaded into the unit specified by *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 hole in the clip. The clip must be loaded for input (IN) or input/output (BOTH).

FF [*unit-name*]

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

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 the unit specified by *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 the same. The clip must be loaded for input (IN) or input/output (BOTH).

FNEW *unit-name* *in* *out*

The FNEW (Insert New Frames) command inserts blank frames into the clip loaded into the unit specified by *unit-name* between the frame specified by *in* (inclusive) and *out* (exclusive). FNEW opens a hole in the clip, and the duration of the clip will increase. The clip must be loaded for input (IN) or input/output (BOTH).

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 the unit specified by *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).

FRM *unit-name* *in* *out*

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

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

GET *unit-name* *dev-type* *control-name-pattern* ...

The GET (Get Control) command retrieves the values of device controls for the unit specified by *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 and 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*"
```

The following example gets the values of the device controls that begin with "vtr.media.clip" for unit U1:

```
GET U1 MED vtr.media.clip*
201 OK
vtr.media.clip.limit.end "*"
vtr.media.clip.preset.start "01:00:00:00"
vtr.media.clip.limit.start "*"
vtr.media.clip.format "movie/vframe"
```

GOTO *unit-name* *timecode*

The GOTO (Goto) command jumps the unit transport specified by *unit-name* to the location specified by *timecode*. The unit transport continues the function that is was executing starting at the target of the Goto command.

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

The STEP (Step) command steps the unit specified by *unit-name* by the number of frames specified by *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.

LIMS *unit-name* *in* [*out* [*direction* [*passes*]]]

The LIMS (Set Edit Limits) command modifies the edit limits being used by the current playback or record function. Which edit parameters may be changed is device-dependent. In general, only the out-point may be changed as a way to stop playback or recording at a specific point even if the unit was cued without a out-point.

LOAD *unit-name* *clip* [*load-mode*]

The LOAD (Load Clip) command loads the clip specified by *clip* into the unit specified by *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).

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* *frame- rate*
- *format* is the format of the clip.
- *size* is the size of the clip content in bytes.
- *resident-size* is the size of the content that is currently resident on the clip cache disk system.
- *start* is the timecode of the first frame of the clip.
- *end* is the timecode of the frame after the last frame of the clip.
- *in* is the current mark-in point of the clip.
- *out* is the current mark-out point of the clip.
- *frame-rate* is the frame-rate (in frames / sec) of the clip. 525- line timing is specified in its approximate form, 29.97.

PAUS [*unit-name*]

The PAUS (Pause) command pauses the unit specified by *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.

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

The PLAY command begins (or resumes) playback on the unit specified by *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.

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 starting recording. For those devices, REC cannot be used to resume recording after a STOP command is issued. The unit must be re- cued.

REVU *unit-name* *in* *out*

The REVU (Review) command performs a edit review 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.

REW *unit-name*

The REW (Rewind) command rewinds the unit specified by *unit- name*. The rewind speed is device-dependent and may be settable with a control.

RHRS *unit-name* *in* *out*

The RHRS (Rehearse) command performs an auto-edit 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.

RSUM [*unit-name*]

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

SET *unit-name* *dev-type* *control1-name* *control1-value* ...

The SET (Set Control) command sets device controls for the unit specified by *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.

The following example sets the value of a storage control for unit U1 so that frames are automatically cleared from the *clip* after being played:

```
SET U1 STOR vtr.storage.fs.clear_after_play true
200 OK
```

SHTL *unit-name* *speed*

The SHTL (Shuttle) command shuttles the unit specified by *unit- name* at the speed specified by *speed*. 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.

SSPD *unit-name* *speed*

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

STON [*unit-name*]

The STON (Standby On) command halts playback but does not stop the unit. 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.

STOP [*unit-name*]

The STOP (Stop) command stops playback or recording.

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

UCLS [*unit-name*]

The UCLS (Unit Close) command closes an opened or created unit. This controller's access will be terminated, and if no other controllers have the unit open, the unit will be deleted.

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

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

- *unit* is the name of the unit.
- *error-code* is the code of the error.
- *error-message* is the textual error message.

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* is the error code of the last error.
- *description* is a textual description of the error.

UFLS [*unit-name*]

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

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 control connection (the UINF returns the identical information for an opened unit).

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

```
*owner* *port-name* *port-mode* *port-physical-name*
```

Where:

- *owner* is the name of the controller that created the unit.
- *port-name* is the name of the media port controlled by the unit.
- *port-mode* is the input/output mode supported by the unit. The possible values are IN, OUT, and BOTH.

port-physical-name is the physical name of the media port controlled by the unit.

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 controller.

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* is the name of the controller that created the unit.
- *port-name* is the name of the media port controlled by the unit.
- *port-mode* is the input/output mode supported by the unit. The possible values are IN, OUT, and BOTH.
- *port-physical-name* is the physical name of the media port controlled by the unit.

UINT [*unit-name*]

The UINT (Unit Interrupt) command interrupts the command currently being processed by the unit specified by *unit-name*. The command will be terminated with error EINTR.

ULOC *unit* *location*

- *unit* is the name of the unit.
- *location* is the current clip location in the format hours:minutes:seconds:frames. In drop-frame mode, the final colon (:) is replaced by a period (.).

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

- *unit* is the name of the unit.
- *control-name* is the name of the control.
- *control-value* is the new value of the control.

ULOC [*unit-name*]

The ULOC (Unit Location) command returns the current transport location for the unit specified by *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*
```

Where:

- *clip-loc* is the clip location, which roughly corresponds to a CTL (control track) timecode.
- *vitc* is the VITC (vertical interval timecode) of the frame.

- *ltc* is the LTC (longitudinal timecode) of the frame.
- *UTC-time* is the UTC time when the unit was 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* is the UST (unadjusted system time) time when the unit was at the specified location. This time can be used to account for application or network latency in time reporting.

UNLD [*unit-name*]

The UNLD (Unload Clip) command unloads the clip currently loaded in the unit specified by *unit-name*. If successful, the response code is 200.

URM *unit*

- *unit* is the name of the unit.

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

- *unit* is the name of the unit.
- *clip* is the name of the loaded clip ("*" if no clip is loaded).
- *status* is the status of the current function (BUSY is the initial state, RUN is the running state, DONE is the completed state, ERR is the error state).
- *function* is 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), REVU (Review), EDIT (Edit), RHRS (Rehearse).
- *location* is the current clip location in the format hours:minutes:seconds:frames. In drop-frame mode, the final colon (:) is replaced by a period (.).
- *speed* is the current playback speed (1000 = normal 1x speed).
- *frame-rate* is the frame-rate (in frames / sec) of the current clip. 525-line timing is specified in its approximate form, 29.97.
- *command-id* is the id of the command currently being processed by the unit.

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 controller.

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

- *clip* *status* *function* *location* *speed* *frame-rate* *command-id*
- *clip* is the name of the loaded clip ("*" if no clip is loaded).
- *status* is the status of the current function (BUSY is the initial state, RUN is the running state, DONE is the completed state, ERR is the error state).
- *function* is 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), REVU (Review), EDIT (Edit), RHRS (Rehearse).
- *location* is the current clip location in the format hours:minutes:seconds:frames. In drop-frame mode, the final colon (:) is replaced by a period (.).
- *speed* is the current playback speed (1000 = normal 1x speed).
- *frame-rate* is the frame-rate (in frames / sec) of the current clip. 525-line timing is specified in its approximate form, 29.97.
- *command-id* is the id of the command currently being processed by the unit.

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.

Once the USYN command has been used to set the default sync 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. The unit specified by *unit-name* waits until the unit specified by *wait-unit-name* reaches the execution status specified by *sync-mode* for the command specified by *wait-id*.

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 the unit specified by *unit-name* according to the synchronization mode specified by *sync-mode* or the default sync mode if the *sync-mode* argument is not provided.

RS-422 Pinouts

The 6-inch mini-DIN8 to male DB-9 adapter cable (SGI part number 018-0650-001) makes the DB-9 end of the cable the same as that of an Origin200 server ttyd1 / ttyd2 for RS-232 only, not for RS-422.

The RS-422 Mini-DIN8 to DB-9 adapter cannot be used with the SGI Media Server.

Table I-1 summarizes the SGI RS-422 pinouts at the DB-9 end of the Macintosh mini-DIN8 to female DB-9 adapter cable.

Table I-1 Apple Macintosh Female DB-9 Pinout

Pin Number	Purpose
2	
3	
4	TX+
5	TX-
6	
7	
8	RX+
9	RX-

Glossary

9-Pin

The Sony protocol for communicating with video devices. 9-Pin is the same as RS-422.

ADAT

The Alesis ADAT Optical I/O interface is an 8-channel, 24-bit digital audio interface. It is important to distinguish between the ADAT Optical format, a 24-bit digital audio I/O protocol, and the ADAT tape decks themselves. SGI workstations support the ADAT Optical interface, in addition to AES digital I/O, either built in or via a low-cost option card.

AES

(Audio Engineering Society) AES Stereo Digital Audio Input/Output. This connection provides a stream of digital audio data that complies with the AES Digital Audio format.

archive system

A repository for storing and distributing digital media data. A StudioCentral 2.0 archive system is one that a Video Server Toolkit has been configured to use. Video Server Toolkit can store *clip*s in, and retrieve them from, a StudioCentral 2.0 archive system only.

asset

Unit of storage in a StudioCentral 2.0 archive system. Each asset consists of descriptive information such as the title and duration; digital media data, if the asset has content; and an index, if required by the content format. For example, each movie, commercial, trailer, thumbnail, and so on, stored in a StudioCentral 2.0 archive system is an asset. Assets may consist of only descriptive information if those assets are used to group other assets.

automation controller

Computers that control broadcast devices. Automation controllers provide features such as playing and recording *clip*s according to a predefined schedule, providing statistics about actual events, previewing schedules, controlling broadcast equipment, and so on.

ATS

MediaHub Asset Transfer Service. The Asset Transfer Service daemon (atsd) can be used to access the meta data and media contents of the Groups and Atoms contained in a MediaHub server without using the full DAS or StudioCentral API's.

Automation Systems

Broadcast Automation Systems (such as the Louth ADC- series) are used extensively by broadcast studios to control multiple video devices. Scheduling of video playback and recording, in conjunction with pre-assigned switcher effects can all be scheduled through a single interface.

CCIR 601

A digital coding standard for television that is applicable to both the NTSC as well as PAL / SECAM technologies. The standard describes the encoding parameters for the 4:2:2 member of the family (where 4:2:2 is the ratio in which the luminance and chrominance sampling frequencies are related). One CCIR 601 video stream (no alpha) is about 22 MB / sec.

clip

The unit of storage in SGI Media Server. Each clip consists of descriptive information such as the title and duration; digital media data, if the clip has content; and an index, if required by the content format.

clip cache

XFS filesystems in which the SGI Media Server stores *clip*s. The clip cache can reside on a normal XFS filesystem that can be shared with other uses, or a real-time filesystem created on a set of striped disks or RAID units.

cue point

Timecodes that are used to move around within a *clip* and to control the portion of the clip that is played. An in-point, the duration, and an out-point are each specified using the *hh:mm:ss:ff* format, where *hh* is the hours, *mm* is the minutes, *ss* is the seconds, and *ff* is the frame number. In drop-frame mode, the final colon is replaced by a period (*hh:mm:ss.ff*).

For example, if a clip with a cue in-point of 00:00:30.00 is cued for playing in the forward direction, it is cued at 30 seconds.

DAS

MediaHub Digital Asset Store.

Diaquest

Deck control software made by Diaquest.

DIF

The compression format used by DVCPRO. A DIF file, for example, is a file containing a clip compressed by DVCPRO.

DIVO-DVC

A Digital Video Option (DIVO) board altered to work specifically with DVCPRO products.

DMBuffer:

Digital Media data transport subsystem. DMBuffers are used for sharing and exchanging time-sensitive visual data between compression devices and algorithms, video input/output, graphics rendering and texturing, and the host processor(s).

DSO

Dynamic shared object. An object that one program can share with another.

DVCPRO

A digital video compression format provided by Panasonic. The format provides 4:1 video compression.

edit point

Persistent timecode values that are stored with a *clip*. Edit points are used to automatically initialize the *cue point*s when the clip is loaded. An in-point, the duration, and an out-point can each be specified using the *hh:mm:ss:ff* format, where *hh* is the hours, *mm* is the minutes, *ss* is the seconds, and *ff* is the frame number. In drop-frame mode, the final colon is replaced by a period (*hh:mm:ss.ff*).

For example, if a clip has an edit in-point of 00:00:30.00, its cue in-point is initialized to 30 seconds when the clip is loaded.

Edit points may also be called “edit marks.”

Fibre Channel

The Fibre Channel Standard defines a high-speed data transfer interface that can be used to connect workstations, mainframes, supercomputers, storage devices and displays. The standard addresses the need for very fast transfers of large volumes of information. Typical high-end Fibre Channel devices carry up to 1 Gb/s data rates.

GRIO

Guaranteed Rate I/O. GRIO is included in IRIX and is used by the SGI Media Server to guarantee disk bandwidth for recording and playback while allowing non-priority access to the clip cache for other operations.

Horita

A company that makes, among other components, timecode readers. Its identifier in configuration files is hsip.

hsip

See Horita.

Little Red

A time code reader made by Miranda. Its identifier in configuration files is littlered.

Louth

The Louth Video Disk Communications Protocol, defined by Louth Automation, provides full-featured control of the Video Server Toolkit using RS-232, RS-422, and TCP/IP.

mcpanel

VTR emulation application for the SGI Media Server.

MediaBase

An application used for browsing/streaming MPEG-1, MPEG-2 (and lower bit rate formats) over intranets, streaming Server on SGI workstations, and browsing clients on all platforms. It is compatible as browsing application for MediaHub and StudioCentral.

media port

Refers to any port through which digital media or control data is passed. That is, the term refers to either a video or a deck control port. See also *port*.

MVCP

Multiple-Unit Video Computer Protocol. A protocol defined by SGI that provides full-featured control of Video Server Toolkit through TCP/IP.

P2

P2 is the Panasonic name for the Sony protocol (RS-422).

port

A point at which an external device connects to the SGI Media Server. Video ports, used by video input and output devices. *DIVO*-DVC video ports are named *DIVO_DVC_n* (for example, DIVO_DVC_2). The term *media port* refers to either a video or a deck control port.

pre-roll

During recording, both source and recording machines can be rewound, generally three frames, so that the beginning of the recording starts when both machines are running at the correct speed. As a result, edits are more frame-accurate.

RS-422

Is the Sony protocol for communicating with video devices.

SMPTE 259M

A broadcast digital video signal standard. All high end digital video formats as well as many peripheral devices have adopted this standard. 259M is a universal standard permitting long cable runs regardless of the size of the facility.

Sony

A media company that uses it's protocol, a standard by default, to control media machines, such as VTRs (RS-422)

StudioCentral

Is a software development environment for building applications and tools for digital asset management solutions. StudioCentral provides the foundation for the acquisition, creation, production, and distribution of digital assets.

unit

A software mechanism that functions like a logical video tape recorder transport and is capable of loading, cueing, playing, and recording digital *clip*s. The unit manages a media *port*, which may be shared with another unit.

A unit may also called a *logical unit*.

Video Server Technology (VST)

Video Server Technology is an element of the *Video Computing Platform* (VCP) from SGI. VST is the clip serving platform element of VCP. The core VST software provides management of a simple database of clips (the Clip Cache), a control API for managing and operating the VST, and a core library which supports various external interface modules.

VDCP

Video Disk Control Protocol (VDCP). RS-422 based protocol used by the Louth Automation ADC series.VDCP is the basis for control protocols for other manufacturers (such as Alamar), as well. VST can parse VDCP through serial ports.

Video Computing Platform

A software platform enabling developers to construct high-performance, scalable video and audio applications on SGI systems.

VTR

A videotape recorder, which is a device that records and plays videotapes.

VST

See Video Server Technology.

Index